



A University of Sussex PhD thesis

Available online via Sussex Research Online:

<http://sro.sussex.ac.uk/>

This thesis is protected by copyright which belongs to the author.

This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the Author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the Author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Please visit Sussex Research Online for more information and further details

A Blockchain and IoT based Framework for Decentralised Smart Campus Environments

By

Manal Alkhamash

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy at the University of Sussex

School of Engineering and Informatics

Department of Informatics

University of Sussex

Brighton

BN1 9QT

October 2022

Abstract

Rapid development in science and information technologies, such as the Internet of Things (IoT), has led to a growth in the number of studies and research papers on smart cities in recent years and, more specifically, on the construction of smart campus technologies. These technologies provide smart solutions or predictions with the convenience of sharing data. However, privacy and security aspects are major concerns because of the data being stored by a third party. Thus, an open research question is how to develop a smart campus that ensures data privacy and security.

This thesis introduces, designs and evaluates a novel smart campus framework architecture based on IoT and blockchain technologies to provide a comprehensive guiding framework. The framework ensures the collection and aggregation of the data from various areas of the campus while increasing the security of the data, thereby providing better service to enhance the experience of the user. The study uses a design science research (DSR) approach to conduct the overall analysis.

As a proof of concept, this framework is tested for integration of blockchain and other relevant technologies into a smart educational environment. Different case scenarios were implemented after determining the blockchain platforms that were the most suitable, based on consensus algorithms. Hyperledger fabric blockchain with a Raft consensus algorithm was used to decentralise a smart educational environment. A benchmark tool and relevant software were used to provide comprehensive results, including execution time, latency and resource utilisation.

The results of this research are particularly useful in the context of developing a smart campus. It is expected that the proposed framework will have useful applications in a variety of fields, where it is necessary to determine whether a satisfactory level of IoT and blockchain technologies has been achieved and maintained in accordance with the relevant safety and security standards for storing and sharing data.

Keywords: Smart campus, Blockchain, Internet of Things, Security, Privacy, comprehensive framework.

Acknowledgement

Today is the day when I finish my journey with my PhD thesis – one of the most challenging and exciting times of my life. I would like to thank the many people who assisted and supported me during the past few years while I was pursuing my thesis. Without their help, I would not have been able to finish.

First, I want to express my gratitude to my supervisors, Dr Natalia Beloff and Prof Martin White for their fruitful guidance, supervision, suggestions and assistance since I started my studies.

Warm and extensive thanks to my awesome husband and gorgeous daughter for their constant support and love during my PhD journey. My deep thanks also to my parents, sisters, brothers and other relatives for their encouragement and for believing in me.

In the same context, my sincere thanks to the government of Saudi Arabia represented by Jazan University for their financial assistance to enable me to finalise my research. Many thanks also to the University of Sussex for providing me this wonderful opportunity to carry out a PhD in a deeply interesting area.

Lastly, I would like to thank all my colleagues and friends for their help and advice through this journey.

List of Publications

Book Chapter:

M. Alkhamash, M. Alshahrani, N. Beloff, and M. White, (2022) “Revolutionising the Approach to Smart Campus Architecture through IoT and Blockchain Technologies,” in *TTBT2021*, Springer International Publishing.

Publications in Conference Proceedings:

M. Alkhamash, N. Beloff, and M. White, (2020) “An Internet of Things and Blockchain Based Smart Campus Architecture,” in *Intelligent Computing*, pp. 467–486.

M. Alkhamash, N. Beloff, and M. White, (2022) “Evaluating Suitability of a Blockchain Platform for a Smart Education Environmnet,” in *Intelligent Systems Conference (IntelliSys)*.

Abbreviations

Term	Description
API	Application Programming Interface
AWS	Amazon Web Services
BFT	Byzantine Fault Tolerant
B2B	Business to business
CA	Certificate Authority
CFT	crash fault tolerance
CLI	Command Line Interface
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
DAPP	Distributed Applications
DLT	Distributed Ledger Technology
DoS	Denial of Service
DDoS	Distributed Denial of Service
DPoS	Delegated Proof of Stake
DSR	Design Science Research
EOV	Execute-Order-Validate
GIS	Geographical Information System
HTTP	Hypertext Transfer Protocol
ICT	Information and communications technology
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ISBN	International Standard Book Number
IPFS	Inter Planetary File System
ISO	International Organisation for Standardisation
MQTT	Message Queue Telemetry Transport

MSP	Membership Service Provider
NFC	Near Field Communication
PBFT	Practical Byzantine Fault Tolerance
PoET	Proof of Elapsed Time
PoS	Proof of Stake algorithm
PoW	Proof of Work algorithm
P2P	Peer to peer
RFID	Radio Frequency Identification
SDK	Software Development Kit
SHA	Secure Hash Algorithms
TPS	Transactions per Second

Table of Content

ABSTRACT	I
ACKNOWLEDGEMENT	II
LIST OF PUBLICATIONS	III
ABBREVIATIONS	IV
TABLE OF CONTENT	VI
LIST OF FIGURES	IX
LIST OF TABLES	XI
CHAPTER 1 INTRODUCTION	1
1.1. RESEARCH CONTEXT	1
1.2. PROBLEM STATEMENT	2
1.3. RESEARCH QUESTIONS	2
1.4. RESEARCH OBJECTIVES	3
1.4.1. <i>General Objectives</i>	3
1.4.2. <i>Specific Objectives</i>	4
1.5. RESEARCH METHODOLOGY	4
1.6. CONTRIBUTIONS TO KNOWLEDGE	6
1.7. THESIS STRUCTURE	6
1.8. SUMMARY	8
CHAPTER 2 LITERATURE REVIEW	9
2.1. SMART CAMPUS	9
2.1.1. <i>Smart Campus Concept</i>	9
2.1.2. <i>Smart Campus Background</i>	10
2.2. INTERNET OF THINGS (IoT)	15
2.2.1. <i>IoT Terminology</i>	16
2.2.2. <i>Challenges of IoT</i>	17
2.3. BLOCKCHAIN TECHNOLOGY	18
2.3.1. <i>Blockchain Terminology</i>	18
2.3.2. <i>Blockchain Components</i>	20
2.3.3. <i>Blockchain Background and Generations</i>	24
2.4. BLOCKCHAIN IN EDUCATION ENVIRONMENTS.....	26
2.4.1. <i>Blockchain-based educational institutions</i>	27
2.4.2. <i>Blockchain-Based Learning Platforms</i>	28
2.4.3. <i>Blockchain-Based Library Management System</i>	28
2.4.4. <i>Blockchain-Based Academic Records and Certificates</i>	29
2.4.5. <i>Blockchain-Based Publishing</i>	30
2.5. SUMMARY	31
CHAPTER 3 ANALYSIS OF SMART CAMPUS ARCHITECTURAL FRAMEWORKS	33
3.1. OVERVIEW OF CURRENT SMART CAMPUS FRAMEWORKS	33
3.2. EXISTING PROBLEMS FOR CURRENT SMART CAMPUS FRAMEWORKS.....	37
3.3. PROPOSED A SMART CAMPUS ARCHITECTURAL FRAMEWORK	40
3.4. THE SMART CAMPUS ARCHITECTURAL FRAMEWORK ANALYSIS	45
3.4.1. <i>Authorisation</i>	46
3.4.2. <i>Trust</i>	46
3.4.3. <i>Privacy</i>	47
3.4.4. <i>Confidentiality, Integrity, and Availability (CIA)</i>	47

3.5. CHALLENGES OF ADOPTING BLOCKCHAIN IN THE SMART CAMPUS ARCHITECTURAL FRAMEWORK	48
3.6. SUMMARY	51
CHAPTER 4 ANALYSIS OF BLOCKCHAIN PLATFORMS FOR SMART EDUCATION ENVIRONMENTS.....	52
4.1. QUALITY REQUIREMENTS FOR BLOCKCHAIN-BASED SMART EDUCATION ENVIRONMENTS	52
4.2. PERMISSIONED VS PERMISSIONLESS BLOCKCHAIN NETWORKS FOR A SMART LEARNING ENVIRONMENT	55
4.2.1. <i>Permissionless blockchain network</i>	56
4.2.2. <i>Permissioned blockchain network</i>	57
4.3. COMPARING CONSENSUS MECHANISMS	60
4.3.1. <i>Compute- Intensive Consensus Algorithms</i>	61
4.3.2. <i>Capability Consensus Algorithms</i>	61
4.3.3. <i>Voting Consensus Algorithms</i>	63
4.4. BLOCKCHAIN PLATFORMS.....	67
4.4.1. <i>Quorum Platform</i>	68
4.4.2. <i>Corda Platform</i>	69
4.4.3. <i>Hyperledger Fabric Platform</i>	69
4.5. RESULTS OF A SUITABLE BLOCKCHAIN PLATFORM FOR SMART EDUCATION ENVIRONMENTS	72
4.6. SUMMARY	72
CHAPTER 5 A SMART CAMPUS FRAMEWORK ARCHITECTURE DESIGN.....	74
5.1. TECHNOLOGIES STACK.....	74
5.2. HYPERLEDGER FABRIC ARCHITECTURE	76
5.2.1. <i>Architecture Components</i>	77
5.2.2. <i>Creating the Fabric Network</i>	79
5.2.3. <i>Transaction Flow in the Fabric Network</i>	82
5.3. OFF-CHAIN STORAGE: IPFS.....	85
5.3.1. <i>Transaction Flow for Integration of Hyperledger Fabric Blockchain and IPFS</i>	87
5.4. SUMMARY	89
CHAPTER 6 A SMART CAMPUS FRAMEWORK ARCHITECTURE IMPLEMENTATION	91
6.1. DEPLOY HYPERLEDGER FABRIC NETWORK.....	91
6.2. LIBRARY MANAGEMENT SYSTEM CASE SCENARIO	94
6.2.1. <i>Smart Contract (Chaincode)</i>	99
6.2.2. <i>API</i>	102
6.2.3. <i>Frontend</i>	110
6.2.4. <i>Dashboard</i>	112
6.3. GENERATING A STUDENT CREDENTIAL REPORT SYSTEM CASE SCENARIO	116
6.3.1. <i>Smart contract (Chaincode)</i>	120
6.3.2. <i>API</i>	123
6.3.3. <i>Frontend</i>	125
6.3.4. <i>Dashboard</i>	127
6.4. EVALUATION	132
6.5. BROADER CONTEXT CASE SCENARIO: DEVELOPING A FRAMEWORK FOR THE ADOPTION OF BLOCKCHAIN IN THE HIGHER EDUCATION CERTIFICATION PROCESS IN SAUDI ARABIA	138
6.6. DISCUSSION.....	142
6.7. SUMMARY	144
CHAPTER 7 CONCLUSION AND FUTURE RESEARCH	146
7.1. THE CONTEXT OF THE RESEARCH.....	146
7.2. RESEARCH QUESTIONS	146
7.3. THE AIMS AND OBJECTIVES	148
7.4. CONTRIBUTIONS TO KNOWLEDGE	148
7.5. SUMMARY OF THE THESIS	149
7.6. RESEARCH LIMITATIONS AND DIRECTIONS OF FUTURE WORK.....	150

REFERENCES	152
APPENDIX..	168

List of Figures

FIGURE 2-1 BLOCKCHAIN STRUCTURE	19
FIGURE 2-2 THE PROCESS OF SIGNING THE TRANSACTION	23
FIGURE 3-1 A NOVEL SMART CAMPUS FRAMEWORK	41
FIGURE 4-1 BLOCKCHAIN NETWORK CLASSIFICATIONS.....	56
FIGURE 4-2 (A) PUBLIC BLOCKCHAIN NETWORK; (B) CONSORTIUM BLOCKCHAIN NETWORK; (C) PRIVATE BLOCKCHAIN NETWORK	56
FIGURE 4-3 DECISION FLOW TO ADOPT A SUITABLE BLOCKCHAIN PLATFORM	59
FIGURE 5-1 TECHNOLOGIES STACK.....	75
FIGURE 5-2 ORDER-EXECUTE ARCHITECTURE	76
FIGURE 5-3 EXECUTE-ORDER-VALIDATE ARCHITECTURE.....	77
FIGURE 5-4 ENDORSER PEER VS COMMITTER PEER	78
FIGURE 5-5 INNER COMPONENTS OF PEER’S LEDGER	78
FIGURE 5-6 HYPERLEDGER FABRIC CHAINCODE.....	79
FIGURE 5-7 CREATING THE FABRIC NETWORK.....	80
FIGURE 5-8 ADDING ORGANISATIONS AS ADMINISTRATORS.....	80
FIGURE 5-9 CREATING A CHANNEL.....	81
FIGURE 5-10 DEFINING PEERS.....	81
FIGURE 5-11 ADDING FABRIC CLIENT APPLICATIONS AND CHAINCODE (SMART CONTRACT).....	82
FIGURE 5-12 COMPLETING THE FABRIC NETWORK	82
FIGURE 5-13 HYPERLEDGER FABRIC TRANSACTION FLOW	85
FIGURE 5-14 TRANSACTION FLOW FOR INTEGRATION OF HYPERLEDGER FABRIC AND IPFS	88
FIGURE 6-1 DEFINITION OF BOOTSTRAPPING THE BLOCKCHAIN NETWORK	93
FIGURE 6-2 LIBRARY MANAGEMENT SYSTEM CASE SCENARIO	95
FIGURE 6-3 A CODE SNIPPET OF THE CREATE BOOK FUNCTION IN THE SMART CONTRACT.....	99
FIGURE 6-4 A CODE SNIPPET OF THE QUERY ALL BOOKS FUNCTION IN THE SMART CONTRACT	100
FIGURE 6-5 A CODE SNIPPET OF THE QUERY A BOOK FUNCTION IN THE SMART CONTRACT	100
FIGURE 6-6 A CODE SNIPPET OF THE ISSUE A BOOK FUNCTION IN THE SMART CONTRACT	100
FIGURE 6-7 A CODE SNIPPET OF THE RETURN A BOOK FUNCTION IN THE SMART CONTRACT	101
FIGURE 6-8 A CODE SNIPPET OF THE DELETE A BOOK FUNCTION IN THE SMART CONTRACT	101
FIGURE 6-9 A CODE SNIPPET OF THE QUERY ALL HISTORY FUNCTION IN THE SMART CONTRACT	102
FIGURE 6-10 A CODE SNIPPET OF THE REGISTER FUNCTION IN API	103
FIGURE 6-11 A CODE SNIPPET OF THE LOGIN FUNCTION IN API.....	104
FIGURE 6-12 A CODE SNIPPET OF THE GET STUDENT FUNCTION IN API.....	104
FIGURE 6-13 A CODE SNIPPET OF THE GET BOOK FUNCTION IN API.....	105
FIGURE 6-14 A CODE SNIPPET OF THE GET REPORT FUNCTION IN API	106
FIGURE 6-15 A CODE SNIPPET OF THE ADD BOOK FUNCTION IN API.....	107
FIGURE 6-16 A CODE SNIPPET OF THE ISSUE A BOOK FUNCTION IN API.....	108
FIGURE 6-17 A CODE SNIPPET OF THE DELETE A BOOK FUNCTION IN API.....	108
FIGURE 6-18 A CODE SNIPPET OF THE ENROL ADMIN FUNCTION IN API	109
FIGURE 6-19 A CODE SNIPPET OF THE REGISTER A USER FUNCTION IN API	109
FIGURE 6-20 A CODE SNIPPET OF THE INITIALISE GATEWAY FUNCTION IN API	109
FIGURE 6-21 A CODE SNIPPET OF THE LOGIN FUNCTION IN FRONTEND	110
FIGURE 6-22 A CODE SNIPPET OF THE REGISTER FUNCTION IN FRONTEND.....	111
FIGURE 6-23 A CODE SNIPPET OF THE ADD BOOK FUNCTION IN FRONTEND	111
FIGURE 6-24 A CODE SNIPPET OF THE ISSUE A BOOK FUNCTION IN FRONTEND	111
FIGURE 6-25 A CODE SNIPPET OF THE RETURN A BOOK FUNCTION IN FRONTEND	112
FIGURE 6-26 A CODE SNIPPET OF THE DELETE A BOOK FUNCTION IN FRONTEND	112
FIGURE 6-27 LOGIN PAGE	113
FIGURE 6-28 SNAPSHOT OF USER DASHBOARD: (A) LIBRARIAN DASHBOARD (B) STUDENT DASHBOARD.....	113
FIGURE 6-29 SNAPSHOT OF THE ADD AND ISSUE A BOOK DASHBOARD.....	114
FIGURE 6-30 SNAPSHOT OF THE BORROW A BOOK DASHBOARD	115
FIGURE 6-31 IPFS DASHBOARD.....	115
FIGURE 6-32 LIBRARY MANAGEMENT SYSTEM CASE SCENARIO BASED THE SMART CAMPUS ARCHITECTURAL FRAMEWORK	116

FIGURE 6-33	STORING AND SHARING STUDENT RECORDS CASE SCENARIO	117
FIGURE 6-34	A SEQUENCE DIAGRAM FOR STORING OR UPDATING STUDENT RECORDS	118
FIGURE 6-35	A SEQUENCE DIAGRAM FOR GENERATING A STUDENT CREDENTIAL REPORT	119
FIGURE 6-36	A CODE SNIPPET OF THE CREATE A STUDENT FUNCTION IN THE SMART CONTRACT	121
FIGURE 6-37	A CODE SNIPPET OF THE QUERY A STUDENT FUNCTION IN THE SMART CONTRACT	121
FIGURE 6-38	A CODE SNIPPET OF THE QUERY ALL STUDENTS FUNCTION IN THE SMART CONTRACT	122
FIGURE 6-39	A CODE SNIPPET OF THE DELETE A STUDENT FUNCTION IN THE SMART CONTRACT	122
FIGURE 6-40	A CODE SNIPPET OF THE QUERY ALL HISTORY FUNCTION IN THE SMART CONTRACT	123
FIGURE 6-41	A CODE SNIPPET OF THE ADD STUDENT FUNCTION IN THE API	124
FIGURE 6-42	A CODE SNIPPET OF THE GET STUDENT FUNCTION IN THE API	124
FIGURE 6-43	A CODE SNIPPET OF THE DELETE RECORD FUNCTION IN THE API.....	125
FIGURE 6-44	A CODE SNIPPET OF THE CREDENTIAL VERIFICATION FUNCTION IN THE API	125
FIGURE 6-45	A CODE SNIPPET OF THE REGISTER FUNCTION IN FRONTEND.....	126
FIGURE 6-46	A CODE SNIPPET OF THE ADD STUDENT FUNCTION IN FRONTEND.....	126
FIGURE 6-47	A CODE SNIPPET OF THE SEARCH CREDENTIAL FUNCTION IN FRONTEND	126
FIGURE 6-48	A CODE SNIPPET OF THE SEARCH CREDENTIAL PAGE FUNCTION IN FRONTEND	127
FIGURE 6-49	SNAPSHOT OF USER DASHBOARD: (A) ADMIN DASHBOARD (B) STUDENT OR THIRD-PARTY DASHBOARD	128
FIGURE 6-50	SNAPSHOT OF THE ADDING A STUDENT RECORDS DASHBOARD	128
FIGURE 6-51	SNAPSHOT OF THE SEARCH PAGE	129
FIGURE 6-52	IPFS DASHBOARD.....	129
FIGURE 6-53	SNAPSHOT OF THE STUDENT CREDENTIAL REPORT	130
FIGURE 6-54	SNAPSHOT OF THE STUDENT INVALID CREDENTIAL REPORT	131
FIGURE 6-55	GENERATING A STUDENT CREDENTIAL REPORT CASE SCENARIO BASED THE SMART CAMPUS ARCHITECTURAL FRAMEWORK.....	132
FIGURE 6-56	HYPERLEDGER CALIPER ARCHITECTURE [243].....	133
FIGURE 6-57	EVALUATING THE FIRST SCENARIO: WRITING TRANSACTIONS MODE ON THE NETWORK LATENCY AND THROUGHPUT	135
FIGURE 6-58	EVALUATING THE SECOND SCENARIO: READING TRANSACTIONS MODE ON THE NETWORK LATENCY AND THROUGHPUT	136
FIGURE 6-59	EVALUATING THE THIRD SCENARIO: WRITING AND READING TRANSACTIONS MODE ON THE NETWORK LATENCY AND THROUGHPUT.....	137
FIGURE 6-60	ACTORS IN DASC [108]	139
FIGURE 6-61	HIGH-LEVEL CONCEPTUAL INFRASTRUCTURE OF DASC [108].....	141
FIGURE 6-62	BROADER CONTEXT CASE SCENARIO BASED THE SMART CAMPUS ARCHITECTURAL FRAMEWORK ..	142

List of Tables

TABLE 1-1 RESEARCH METHODOLOGY ACTIVATES, BASED ON PEFFERS ET AL. DSR MODEL.....	5
TABLE 4-1 SUMMARY OF BLOCKCHAIN NETWORK TYPES WITH THEIR FEATURES	58
TABLE 4-2 SUMMARY OF THE COMPARISON OF CONSENSUS ALGORITHMS FOR AN EDUCATION ENVIRONMENT	67
TABLE 4-3 SUMMARY OF THE COMPARISON OF BLOCKCHAIN PLATFORM FOR EDUCATION ENVIRONMENT	71
TABLE 6-1 ENVIRONMENT CONFIGURATION.....	92
TABLE 6-2 TESTING THE WRITING TRANSACTIONS MODE	134
TABLE 6-3 TESTING READING TRANSACTIONS MODE	135
TABLE 6-4 TESTING WRITING AND READING TRANSACTIONS MODE	137

Chapter 1 Introduction

This chapter presents an overview of the conducted research and context. Then it describes the research problem followed by the research objectives and questions that this thesis focuses on. Finally, it highlights the contribution to knowledge.

1.1. Research Context

Information and communications technology (ICT) development is a never-ending process, which has led to a growth in the number of studies and research papers on smart cities in recent years. The concept of a smart city is not only about constructing traditional infrastructure, such as a transportation system, but also involves ICT infrastructure in order to improve quality of life and enhance the profile of the city [1]. Globally, governments are working towards digitalisation with the development of smart cities to accommodate the projected rise in an average city's population. From 2015–2050, the population in a city is expected to grow from 55 per cent to 66 per cent, which will require the optimisation of resources and the use of technologies, such as IoT, to implement a sustainable and intelligent environment [2] for use with governance, public transport, traffic, etc. Therefore, the term 'smart city' can be generally defined as dynamically integrating the physical and the digital worlds, in which different data resources are automatically gathered in real time [1], [3], [4]. By utilising high-speed networks, the changes in the physical world can be captured and transferred to data centres so that they can be stored and processed [5]. This means that, in order to capture the necessary data, there needs to be significant numbers of sensors at diverse locations that can capture this 'big data'. In addition, the Cloud needs to be utilised in order to store and analyse the data.

Consequently, there are many areas that can be developed under the smart city framework to achieve the overall goal of improving citizens' quality of life. There have been many contributions and research papers in different areas to develop smart systems, such as medical and health care [6]–[9], supply chain management [10], [11], traffic [12], [13], and education systems [14]–[16] that together can build smart cities.

Since a smart campus constitutes an essential element of a smart city, and the concept of the smart campus comes from the notion of smart cities [17], many researchers have focused their attention on developing smart campuses, trying to address the topical question of how to develop an intelligent campus by contributing the same ideas and bases of intelligent cities [18]. Therefore, the aim of this thesis is to study different technologies and to design a novel architecture for a smart campus in order to develop an intelligent campus. Such a smart and intelligent campus architecture or framework is likely to exploit the IoT, blockchain, and smart contracts as part of its many technology solutions.

1.2. Problem statement

The rapid development in advanced technologies, such as IoT and blockchain, has led to a growth of interest in a number of application areas, including smart environments such as the smart campus, which can be considered as an example of a smart city. Despite such growth, no comprehensive guiding framework has been developed for emerging IoT and blockchain technologies deployment in the smart campus environment, particularly in relation to security and privacy aspects, as well as to the mitigation of known problems with IoT and blockchain in existing applications. In addition, the current smart campus systems rely on centralised architecture that is developed using a centralised server to manage and control connected nodes which suffers from various threats and complications that will be discussed in Chapter 3. Therefore, this thesis proposes a novel architectural framework for the IoT and blockchain applications deployed within a smart campus environment and comparing the main technologies involved. The framework also combines several innovative technologies, including infrastructure, storage and communication technologies to design various campus service systems, such as smart systems for learning, library, building, parking, waste and water management, etc. In addition, different case scenarios are implemented to demonstrate the performance and the applicability of the framework.

1.3. Research Questions

The main aim of this research is to propose and evaluate a comprehensive framework for a smart campus implementation focusing on security aspects. To achieve this objective of this study, it is necessary to answer the following research questions:

- **What technologies, including hardware and software, contribute to the building of a highly technological smart campus?** An extensive literature review is conducted to understand the analysis of current innovative research relevant to a smart campus and the analysis of existing smart technologies and intelligent software/hardware system undertaken in different areas of a campus, such as learning and teaching, building management, waste management, energy management, water management, transportation, security, etc.
- **How can the smart campus technologies be characterised?** A wide variety of architectures for a smart campus have been proposed with several goals. Most existing architectures for a smart campus generally rely on advanced technologies, such as IoT, which allow devices to interact with each other and analyse the acquired data for various purposes. These architectures are studied in more details to propose a smart campus architectural framework.
- **What methods can be used to develop a comprehensive framework for a smart campus implementation?** Systematic comparative analysis is used to investigate the best technologies for implementation, including selecting blockchain platform, consensus algorithm, and off-chain storage. Then several use case scenarios of smart educational environment are modelled and evaluated as proof of concept.
- **What are the implications on the applicability of a comprehensive framework for a smart campus in the future?** To answer this question, a novel framework that combines several innovative technologies and tests these technologies in a higher education certificating system for issuing authentic, verifiable and sharable student credentials has been develop. A collaborative book chapter with another PhD student named Mona Alshahrani has been prepared to answer this question.

1.4. Research Objectives

The research has several objectives that can be presented as the follows:

1.4.1. General Objectives

- Propose and evaluate a comprehensive framework for a smart campus implementation.

- Ensure authorisation, trust in addition to the CIA triad of Confidentiality, Integrity, and Availability for the proposed framework data are main aspects in privacy and security requirements.

1.4.2. Specific Objectives

- Gain knowledge on related research work for a better understanding on smart campuses' principles and concepts and determine their technologies.
- Investigate the main features of emerging IoT and blockchain technologies to deployed to the smart campus environment to extract the domain knowledge.
- Analyses the validity of distributing the smart campus applications and systems and gain new insights.
- Implement choosing different case scenario to evaluate the applicability of a novel architectural framework.

1.5. Research Methodology

The research follows a Design Science Research (DSR) approach to conduct the overall study. The DSR approach has recently gained traction in the discipline of information systems research generally and in the current study particularly for the following reasons:

- The approach's fundamental principle is to solve existing research problems and improve the current state of knowledge by creating an artefact [19]. The artefact can lead to many different output forms, including design principles, constructs, methods, design theories, models, technological rules or any type of knowledge considering contributions [20]. In this research, the artefacts are proposed and evaluated for a comprehensive guiding framework for the implementation and development of a smart campus.
- This approach provides a transparent roadmap and detailed guidance about artefacts, methods and roles. Hence, a study can produce outcomes that are considering relevant and rigorous [21]. Therefore, the approach can be used to apply analytical and synthetic perspectives and methods to present the profound study.

- DSR can address problems faced by information system researchers in any stage.

This research follows the DSR process model that adheres to the guidelines of Peffers et al. [19], which consist of six activities or steps of: the identification of the research problem, which focuses on defining and establishing the research problem to pursue a solution; the objectives of a solution, which infers of possible solution for the previously defining research problem; the design and development, where the solution artefact is determined and created; the demonstration which aims to demonstrate the efficiency of the solution artefact for solving the research problem; the evaluation is where the artefact is measured and observed about how it can support the research solution; and communication, the final phase, where the problem and the artifact's design and efficacy are shared with relevant researchers and audience.

Table 1-1 describes how these activities have contributed to this research with a brief description.

Table 1-1 Research methodology activates, based on Peffers et al. DSR model

Activity Title	Description
1- Identification the research problem	During this step, the problem of the current deployment of the smart campus environment, particularly in relation to security and privacy, was defined, and the mitigation of known problems with IoT and blockchain in existing smart campus applications was determined.
2- Objectives of a solution	In this activity, the objectives of proposing the architecture that incorporates and exploits blockchain technology to overcome some of the prevalent issues for current applications and the implementations of different smart campuses' areas were inferred.
3- Design and development	In this stage, a novel architectural framework for the IoT and blockchain applications were designed within a smart campus environment, comparing the main technologies involved.
4- Demonstration	During this phase, the educational smart environment was implemented to demonstrate a proof of concept and the utility of the artefact.
5- Evaluation	In this stage, the proof-of-concept case scenarios was evaluated by using external software, and the results reported.
6- Communication	During this study, several academic papers have been published, and the researchers attended international conferences to communicate the research to the relevant scientific community.

1.6. Contributions to Knowledge

The main aim of this thesis is to propose and evaluate a comprehensive framework for a smart campus implementation, and this goal provides several contributions to the knowledge which can be summarised as follows:

- 1 Analysis of the current innovative research relevant to a smart campus and of existing smart technologies and intelligent software/hardware system undertaken in different areas of a campus, such as learning and teaching, building management, waste management, energy management, water management, transportation, and security;
- 2 Analysis of the best technologies including consensus algorithms, blockchain platforms and off-chain storages to use in different case scenarios as proof of concept;
- 3 A novel, comprehensive smart campus architectural framework, using blockchain technology to handle the issue related to a centralised IoT technologies;
- 4 Developing a systematic analysis method to determine which blockchain platform and consensus algorithms are suitable for blockchain-based smart education environments;
- 5 Evaluation of the framework by measuring performance and security requirements; and
- 6 Testing of the future applicability of the comprehensive framework for a smart campus in a boarder context (student certification scenario).

1.7. Thesis structure

This section presents a structured and brief description of this thesis, which consists of seven chapters.

Chapter 1: This chapter presents an overview of the study context and describes its problem. In addition, it provides the research aims and objectives as well as the research questions. The chapter also presents a brief list of the research contributions to knowledge.

Chapter 2: This chapter presents a literature review on smart campuses. This includes the concept of the smart campus and is followed by a study of each campus area in more detail with their current research. It then discusses recent innovative studies on smart campuses to

understand how the smart campus is characterised and to identify the most well-thought-out and effective approaches, ideas and developed hardware and software systems for smart campuses. The chapter also reviews blockchain technology including its definition, terminology and a brief history. Furthermore, the chapter covers the use of blockchain technology in various systems and applications in smart educational environments.

Chapter 3: This chapter analyses the various components of a smart campus, including architectures, platforms, and technologies with their limitations. In addition, it proposes a novel architectural framework for the IoT and Blockchain applications deployed within a smart campus environment, comparing the main technologies involved. Then, with particular consideration, it discusses security and privacy requirements.

Chapter 4: This chapter acts as a guide in selecting and developing a suitable blockchain platform for experimenting with educational applications, such as students' transcripts, certificates, credentials, or any other accomplishment data forms. Determining which blockchain platform to use requires discussing the quality requirements for blockchain-based smart education environments. Then this chapter reviews current and well-known consensus algorithms and provides a comparative analysis of all the platforms to choose a suitable one, according to the smart education software requirements.

Chapter 5: The architecture and design of a novel framework is explained in this chapter. It provides an overview of the technologies stack required for the framework and the reasons for on-chain and off-chain data storage. Then, each on-chain and off-chain technology used is discussed independently with more detail regarding its components, design, and transaction flow.

Chapter 6: The chapter presents the use of previous architectural design by implementing different case scenarios in smart campus, particularly a smart educational environment, as a proof of concept. In addition, it provides the testing environment with dissection to evaluate the framework.

Chapter 7: This chapter presents an overview of the whole study and summarises the outcome of this thesis. In addition, it provides the resulting recommendations and the research limitations. It also discusses the directions for further research.

1.8. Summary

This chapter introduced the context of this research and outlined the importance of developing smart environments, particularly smart campuses. However, security and privacy concerns have become more prevalent due to the increasing number of connected devices. Therefore, there is a need to propose a new smart campus framework that combines the advantages of both the IoT and blockchain technology and that can be used as guide to develop various campus services, such as smart learning, smart building, smart parking etc. This chapter presents the research goal with its objectives, methodology, contribution to knowledge and the structure of this thesis. The next chapter will address the concept of the smart campus and study each campus area in more depth, including the technology used, the limitations and blockchain technology.

Chapter 2 Literature Review

This chapter covers the theoretical background and literature review of this thesis. It includes two parts to answer the first research question: ‘What technologies, including hardware and software, contribute to the building of a highly technological smart campus?’. The first part provides an overview of recent innovative studies on smart campuses. The section starts by addressing the concept of the smart campus, followed by studying each campus area and their current research to understand how the smart campus is characterised and find the current research gaps.

In the second part of this chapter, blockchain technology will be explained in more detail. The term ‘blockchain’ will be defined first as a fundamental concept alongside studying the history and previous research on the subject, followed by clarifying blockchain’s essential components, such as smart contract, digital ledger and consensus and the way they work together. Furthermore, the section will address the different blockchain generations since their structures vary. In addition, this section will cover the use of blockchain technology in various systems and applications in smart educational environments while determining the gaps in recent research.

2.1. Smart Campus

This section presents the smart campus concept and an overview of its current publications and related areas. The research background is used to help to identify and classify diverse studies with a broad range of introduced smart campuses, proposed approaches and notions, developed hardware and software systems, designed technical platforms, etc.

2.1.1. Smart Campus Concept

Traditionally, a campus can be defined as a land or an area where different buildings constitute an educational establishment. A campus often includes classrooms, libraries, student centres, residence halls, dining halls, parking, etc. Nowadays, campuses have adopted advanced technologies, such as visual learning environments [22], [23] and timetabling systems [24], [25] in order to provide high-quality services for stakeholders (e.g. academics, students,

administrators, and services functions) on campus and to monitor and control facilities. These developments should be evolving constantly in order to increase efficiency, cut operational costs, reduce effort, lead to better decision-making, and enhance the student experience [26]. Thus, the term ‘smart campus’ can be defined as a place where digital infrastructure can be developed and that has the ability to gather information, analyse data, make decisions, and respond to changes occurring on campus without human intervention [26], [27]. The authors in [28] define a smart campus as an environment where the structure of ambient learning spaces – application context based on virtual spaces – integrates social and digital services into physical learning resources. If we think of a smart campus as a holistic framework, it encompasses several themes, including but not limited to automated security surveillance and control, intelligent sensor management systems, smart building management, communication for work, cooperation and social networking, and healthcare. Several innovations have been proposed for smart campuses, ranging from developing a whole framework using technologies, such as mobile technologies, blockchain, the IoT, and the Cloud to assist learning to enhancing security systems utilising technologies, such as ZigBee and radio frequency identification (RFID) [29]–[32].

2.1.2. Smart Campus Background

Many studies and architectural plans with different goals have been undertaken on the subject of the smart campus [33]. This smart campus research largely breaks down into the following areas: teaching and learning, data analysis and services, building management and energy use on campus, campus data mining, water and waste management use on campus, campus transportation, and campus security.

2.1.2.1. Smart Campus Learning Environments

Much research has been focused on constructing smart campuses by developing suitable technologies and applications that involve teaching and learning. Therefore, the common purpose of designing and developing a smart campus has often been from a learning and educational perspective. Ng et al. [31] developed a novel holistic environment for a smart campus known as iCampus. The aim of their research is to propose a beginning-to-end lifecycle within the knowledge ecosystem in order to enhance learning. Atif and Mathew designed a framework for a smart campus that integrates a campus social network within a real-world educational facility [34]. The study’s goal was to provide a social community where knowledge could be shared between students, teachers, and the campus’s physical resources. Further, [1]

proposed a model of a smart campus to enable stakeholders on the campus to shape and understand their learning futures within the learning ecosystem. Based on cloud computing and IoT, [35] stated the concept of a smart campus and demonstrated some issues that related to intelligence application platforms after establishment. However, these approaches were focused only on proposing a smart campus by using IoT technology which suffers from various threats and complications as will be discussed in section 3.2.

2.1.2.2. Smart Campus Data Analysis and Service Orientation

Other research has considered the development of smart campuses based on data analysis. According to [36], a smart campus should be able to gather data from a crowd and analyse it by using crowdsourcing technologies in order to deliver services of added value. In 2011, [37] explained the prototype of a smart campus implementation that uses semantic technologies in order to integrate heterogeneous data. However, some researchers have envisioned smart campuses from social networking aspects. For instance, [38] elaborated upon an architectural system that can be deployed on campus in order to support social interaction by using service-oriented specifications. This will depend upon their proposed social network platform (WeChat) and an examination of its architecture, functions, and features. Xiang et al. developed a smart campus framework based on information dissemination [18]. However, these approaches did not address blockchain technology in order to eliminate centralisation.

2.1.2.3. Buildings Management and Energy Efficiency on Smart Campuses

Several of the current initiatives that are developing smart campuses have been based on high-energy efficiency perspectives. In order to decrease the energy consumption of buildings, monitoring and controlling environmental conditions is essential, such as controlling both natural and artificial lighting, humidity, and temperature. An example of this is a project that was undertaken at the University of Brescia in Italy in 2015 that aimed to enhance energy efficiency inside buildings by monitoring lighting, temperature, and electrical equipment by using control systems, automation, and grid management. The project progressed in stages towards this goal. First of all, it aimed to reduce the consumption of the buildings' energy by analysing possible actions.

Then it attempted to implement different measures and evaluated their efficiency. Simultaneously, in order to enhance users' awareness of energy consumption, a system for monitoring operational conditions was also developed. Finally, the project evaluated the energy balance between consumption and generation, renewable energy production, and energy

reduction. The outcome displayed a significant energy consumption reduction of 37.3% while improving the buildings' thermal properties [39].

In addition, [40] proposed and implemented a web-based system to manage energy in campus buildings known as CAMP-IT. The system aimed to optimise the operation of energy systems in order for buildings to achieve goals of reducing energy consumption while at the same time enhance the quality of the indoor environment in terms of visual comfort and air quality. The modelling collected, controlled, and analysed the energy load for each building and for the campus as a whole. The results showed a reduction in energy consumption of nearly 30%.

A smart building, which includes a sensor-controlling unit, security unit, energy provider and control unit, helps to produce and maintain a safer, comfortable and productive environments in the most cost-effective way possible. The above devices communicate using centralised and controlled architectures. All data have to be stored in a central operations unit to make decisions for energy consumption, which involve a lack of efficiency, security and scalability, as will be discussed in more details in Section 3.2. Alternatively, decentralising these architectures could enhance the system security.

2.1.2.4. Smart Campus Data Mining

Additionally, some researchers have focused on applying interest mining, which is based on location, context awareness, proximity, and user profiles as well as other related information, to assist users in meeting their needs within the campus environment. In 2014, [41] studied web log mining, which is an essential technique in web data mining to determine users' characteristic interests by developing a reliable and efficient method of data pre-processing. In 2010, [42] proposed a data-mining method from e-learning systems to identify users' interests and obtain information about learners' logs and knowledge background. Along these lines, the model would be able to automatically recommend resources that may be of interest to individual students. However, protecting user profiles and preferences from leaking or hacking should be considered, as they are considered sensitive data.

2.1.2.5. Water Management on Smart Campuses

Regarding water and waste management, since they are considered expensive and important services on smart campuses, several research studies have focused on proposing management systems on campuses for these services in order to reduce the environmental and financial

impact [26]. In terms of sustainable water management, there are three important pillars: water harvesting processes, water recycling, and water consumption reduction [43].

Different approaches have been proposed to manage water. Some focused on controlling and monitoring the water level and water consumption on campus. For instance, [44] developed a water monitoring system to reduce water consumption on campus. The system designed a three-dimensional map of the campus and used a geographical information system (GIS) to display a water pipeline in the electronic map with detailed status information in real time. Therefore, the model can monitor water directly from pipelines; detect any problems that occur in the equipment, such as leaking; and analyse the amount of water consumption.

In 2015, [45] developed a suitable system for medium-sized campuses to monitor the water balance in real time. The design used an ultrasound level sensor, a cloud software stack, and communication links and carefully considered industrial design. To be able to monitor the water, the system measured the water level in tanks by sending ultrasound pulses to the water's surface. After observing the reflection, the sensors can estimate the distance and calculate the tank volume. Based on previous work, [46] developed an automatic water distribution system for large campuses so that each tank on the campus would have enough water to meet the local needs. The authors utilised ultrasonic ranging sensors, which are suitable for measuring water levels in large tanks, and a wireless network using sub-GHz radio frequency to connect sensors across long distances for further analysis.

Moreover, many other experiments have proved efficient for developing water management systems, and they can be implemented on smart campuses to reduce water consumption [47]. For example, [48] developed a meter of a smart water that can provide a user with real-time reading information, analyse his consumption data, and present it in visual graphs to improve the readability. Simultaneously, the system monitors the consumption and alerts the user if there is unusual water usage. Therefore, the main goal of a water management system is water conservation by maintaining appropriate traceability of a water supply. This requires storage resources and high processing power. However, these approaches have not been discussed in light of the security mechanisms that should be used to protect the recorder data from being manipulated.

2.1.2.6. Waste Management on Smart Campuses

Similarly, numerous studies have been devoted to developing waste management systems. Authors in [49], [50] stated that general research studies in this area focused on developing

waste tracks and bins with sensor devices attached to collect and analyse real-time data. This information can be used for several purposes, for example, for developing an efficient cleaning timetable and preventing overfilling of bins. Ebrahimi et al. [51] in 2017 investigated the current waste and recycling infrastructure on Western Kentucky University campus to determine whether it had an adequate service by using spatial techniques, such as GIS, to track, recognise, and visualise waste and recycling bins in a large-scale area. They used spatial information for analysis and decision making to reduce solid waste stream and improve the university's recycling stream. Furthermore, they drew an accurate roadmap for a suitable waste management plan for the campus. Although most papers use different techniques for waste management systems on smart campuses, they are still at the primary stages, and they lack a generic model. According to Ahmad et al. [52], effective waste management needs close cooperation and coordination among relevant participants such as waste source owners, waste disposal facilities, collectors and shippers. However, the current waste management systems face a number of challenges due to the lack of sharing adequate waste data among the participating nodes. Because blockchain technology has traceable feature as well as share data in secure, effective and verifiable manner, it could be used to enhance waste management systems.

2.1.2.7. Smart Transportation

Recently, global positioning system (GPS) has become the most common method for streaming a location and tracking a moving object, such as a vehicle on the road. To improve the accuracy of GPS, external information is needed, such as Wi-Fi, digital imaging, and computer vision [53]. The authors in [54] developed a tracking system for buses using GPS devices that reported the buses' locations every ten seconds. The location was sent from the server via SMS. The system also had safety features, such as the ability to send alerts or emergency reports when the vehicle crashed or was stolen. Other studies have tracked the location of a college bus using a mobile phone and Google Maps [55], [56]. Saad et al. [53] developed a real-time monitoring system for a university bus that used a GPS service to send the location of the bus to a cloud database every second. The system could also analyse data to estimate the bus's arrival time. Generally, smart transportation and vehicles provide a range of services and applications. However, communication among devices is over a wireless medium that is vulnerable to various cyberthreats [57]. For examples, adversaries can manipulate information communicated between different nodes. One possible solution to solve this data security issue is to integrate blockchain platform to such environments to prevent unauthorised modification

of data and allow sharing of data among trusted peers, such as IoT devices, organisations and individuals.

2.1.2.8. Smart Security

Many mobile applications have been developed for campus safety. Some of them allow users to contact campus security guards, such as EmergenSee and CampusSafe, whereas others, such as Guardly and CircleOf6, allow friends to contact each other [58]. These applications allow user location, photos, and situation descriptions to be shared with campus security guards.

In addition, [59] also proposed a smart campus framework that includes several aspects, of which security was a notable one. They pointed out that a smart system can reduce burglaries by detecting glass breaking or any distinct sound; then, the system has the ability to alert security to the location. Also, the system may have the ability to reduce drug or alcohol abuse by alerting public safety to the presence of alcohol.

Therefore, a smart campus can be described as an environment that has the ability to provide a suitable infrastructure in order to deliver services required in light of contextual awareness. In addition, it is a well-structured place that can generate huge amounts of information to a number of users by using their profiles and locations in order to best address their needs. Consequently, the desirable characteristics of a smart campus are accurate context awareness and ubiquitous access to networks, efficient and optimal utilisation, many varied resources, and the use of objective principles as a basis to make smart decisions or predictions.

2.1.2.9. Summary

All the above approaches and implementations are useful and contribute to build smart campuses; however, they rely on IoT technologies with a centralised system architecture, which lead to many issues and will be discussed in the next chapter with the proposed architecture that incorporates a distributed architecture exploiting blockchain technology to overcome some of the prevalent issues. Next, IoT technology is introduced in more details including its terminology and challenges.

2.2. Internet of Things (IoT)

Due to the continual growth of the Internet, connectivity and communication are ubiquitous. The integration of humans and devices has enabled connection to the Internet and data transfer

automatically, which has created the so-called Internet of Things (IoT). IoT technology facilitates the global connectivity of computer networks, enabling the remote control of various 'smart' objects to access specific services. IoT is an innovation that combines digital and physical components to enable novel business models and the creation of new products. With the efficient increase in broadband communication and power management, along with advances in microprocessors and increasingly reliable memory, it has become possible to digitalise functions and environments. Therefore, creating a smart world and the information thus generated may prove useful in various service domains, including smart cities, smart homes, and smart campuses.

This section will explain IoT technology in more detail. It begins by assessing IoT technology terminology from different perspectives and identifies the one this thesis follows.

2.2.1. IoT Terminology

Over the years, researchers have defined IoT from different perspectives [60]. Some definitions have concentrated on the objects that connect to networks; others have focused on aspects related to IoT such as network technology and network protocols. Some descriptions of IoT focus on semantic challenges such as big data information, storage, and search. The European Commission defined IoT as an approach to developing smart environments through merging physical and virtual worlds [61]. The term IoT usually refers to a system of smart devices connected to the Internet with the ability to identify themselves and communicate with each other by collecting and sending data via the network [62]. Thus, it allows people, things, and processes to communicate and be connected in any place, at any time, using any service, and with any network.

From a technological perspective, when devices embedded with sensors and chips such as tags, near-field communication (NFC), and radio frequency identification device (RFID) are to be identify, control, and manage the devices, they use IP addresses and other communication protocols such as Message Queue Telemetry Transport (MQTT) to communicate without human interfaces [63]. When devices connect to the Internet, they can start transmitting data to computing technologies such as the cloud, which is a powerful platform that integrates data analytics tools, data storage, and data delivery models to perform services for users and businesses. Once the data are received in the cloud, they are processed by relevant software and, in turn, valuable information is sent to the client. These data create a smart environment when they are used to make correct decisions, detect problems before they occur, and save time

and money. Therefore, major IoT characteristics are perception, network, and intelligent processing.

2.2.2. Challenges of IoT

With the growing business requirement for service applications, there is a need to develop more devices and emerging technologies to ensure availability anytime, anywhere, as well as to propose protocols to solve compatibility issues and integration among heterogeneous objects that are connected to the network. With these developments, IoT has faced several challenges in terms of security and privacy. To solve these issues, it is necessary to revise the traditional IoT architecture, which consists of three basic layers: the perception layer, network layer, and application layer [64]. For example, many developments have occurred in terms of security measures in the perception layer, including adding access control for the connected devices (i.e., to protect privacy) and providing different encryption mechanisms (i.e., to encrypt the signal from electronic tags). In addition, the network layer has been altered to enable more dynamic topologies, end-to-end authentication, security routing, and the key agreement process, among other features. One example is developing the IPSec protocol and IEEE 802.15.4 standard protocol to provide secure channels of communication between two devices. In the application layer, many mechanisms have now been developed to reduce the leakage of confidential information, as well as to bolster the robustness and security of information management (e.g., password management and resource management).

Nowadays, the amount of the connected devices is assumed to be around 75 billion in 2025 which is rapidly increases three times than in 2019 [65]. The exceptional growth in the system of IoT has created new opportunities by which methods allow information to be shared and accessed easily. Reyna et al. [66] highlights the cause of such initiatives as mainly the existence of the open data paradigm. These creative systems and methods face some significant vulnerabilities, such as a shortage of confidence and the leaking of data. As stated by Zheng et al. in [67], Internet of Things (IoT) is considered as one of the leading field in adopting blockchain technology. The blockchain can increase the efficiency of the IoT by providing a sharing service that is trusted, where we can benefit from a decentralised environment, in which the information is easily traceable and reliable. Using blockchain technology integrated with IoT will increase security whereas in any point of time the data's sources can be recognised with guarantee the data immutability. As result of this integration, the IoT will provide secure environment where information can be securely shared between several participants [66]. This

research will contribute to improving campus IoT security by merging it with blockchain technology, as discussed through this thesis.

Next, blockchain technology is introduced in more details including its components, the way they work, and an overview history of the technology.

2.3. Blockchain Technology

Recently, blockchain technology research has become a trend in computer science, with growing attention from researchers and organisations. Since 2008, it has ranked among the top five technology trends and is considered the next revolution in technology as it provides solutions to the issues related to classical centralised architecture [68], [69]. Briefly, it acts as a distributed database recorded in blocks. Blocks of data are secured against revision or change, leading to increased security in the network [70], [71].

This section will explain blockchain technology in more detail. It begins by assessing blockchain technology terminology from different perspectives and identifies the one this thesis follows. Next, blockchain components will be presented to explain the main features and mechanisms. After that, the history of the technology is delineated by explaining the different blockchain generations and clarifying to which generation the thesis belongs.

2.3.1. Blockchain Terminology

Narayanan et al. [72] indicated that the term ‘blockchain’ does not have a specific or standard definition. However, the term has described systems that resemble Bitcoin. There are various definitions of blockchain. For example, Coinbase is a cryptocurrency exchange, one of the largest platforms in the world, that provides blockchain as ‘a distributed public ledger that contains the history of every bitcoin transaction’ [73]. Coinbase clarifies blockchain from the perspective of digital currencies, while this technology is usable beyond financial applications.

Other definitions focus on the primary technical components, such as transactions, digital ledger and cryptographic verification. For example, Sultan et al. [71] indicated that blockchain is ‘a decentralised database containing sequential, cryptographically linked blocks of digitally signed asset transactions, governed by a consensus model’. Kaal’s and Dell’Erba [74] defined blockchain as ‘... a shared digital ledger or database that maintains a continuously growing list of transactions’.

Some definitions describe blockchain more broadly. Buterin [75], the founder of the Ethereum blockchain, one of the most common platforms, defined the technology as ‘a decentralised system that contains some kind of shared memory’. According to the National Institute of Standards and Technology, ‘blockchains are tamper-evident and tamper-resistant digital ledgers implemented in a distributed fashion and usually without a central authority’ [70]. This detailed description of blockchain covers a broader overview. This thesis will follow such a definition. It describes blockchain’s vital feature as distributed technology with an immutable and decentralised structure. The term ‘blockchain’ is applicable in different ways. For example, it can describe a suite of technologies, a data structure and an algorithm’s name. In addition, the term can describe distributed peer-to-peer systems as an umbrella term. In this thesis, the term refers to all interpretations of the blockchain.

Therefore, Blockchain represents a distributed ledger that provides a transaction within a decentralised digital database. The transaction is verified and agreed upon by a network of computers before it is added and updated to the ledger. Blockchain allows parties to exchange assets in real-time without going through intermediaries [76]. A block in the ledger formed from a blockchain contains a transaction or a grouped set of transactions. Each block is linked to the previous block using the hash function and timestamp (see Figure 2-1). The first block differs from the other blockchain blocks and is known as a genesis block [77]. The block is hardcoded at implementation because it has unique characteristics, does not point to a previous block and does not hold any transactions. However, it contains details of network configuration.

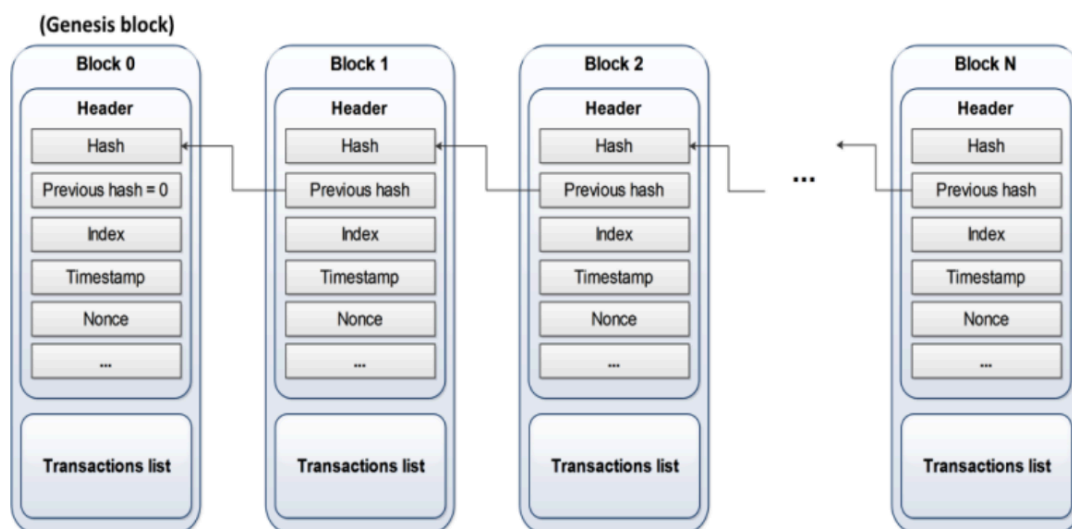


Figure 2-1 Blockchain structure [77]

According to the above figure, the blockchain data structure shows the idea of a link list data structure as each block is referred to the preceding block. Both have a genesis element in the first part of the list, with no ancestor. However, noteworthy differences exist between them. Linked list data structures store data in a linear structure and use a pointer function to refer to the previous data. The linked list can be added to or deleted from data anywhere in the list [78], with no need for validators to check the update. The unique feature of the blockchain is that it is immutable and decentralised. A block in blockchain cannot be modified or manipulated since each block uses a hash function referring to its preceding block, and each block has a unique hash [79]. The hash function is a crucial algorithm that helps to validate and write the data into a blockchain. Each peer in the network has a copy of the blockchain ledger, and any changes in the network mean that the ledgers are synchronised and updated.

The following section explains the principal technical blockchain technology components in more detail.

2.3.2. Blockchain Components

At a high level, numerous computer science mechanisms have helped to develop blockchain technology, such as the hash function and asymmetric-key cryptography. There are also components, such as blocks, distributed ledgers, consensus and transactions, that work together to build a secure and robust platform. Each component is explained as follows:

2.3.2.1. Blocks

A block is a crucial element in blockchain technology as data or transactions form in blocks. Each block links to the previous one to form a chain of blocks [80]. Blockchain uses the hash function to link the chain, which provides a unique signature for the block. If the transaction inside the block has been altered, the entire signature will change even if the data is altered one bit. This method helps to detect the altered block.

Before explaining the data fields of a block, it should be noted that in the implementation stage, each blockchain platform can initialise its own data fields. However, it is typical that each block in blockchains consists of header, data and metadata, as in the following:

- **Block header:** This consists of the block number in the blockchain. In some blockchains, it is referred to as block height. The numbering system uses integers starting from 0, which is the first block and also known as the genesis block, and for each appended block the

number increases by 1. The block header also has the header hash value of its previous block as well as its current block hash, which represents a hash of all data blocks.

- **Block data:** This data has all the transactions grouped in the block. Each transaction contains the header, which has the transaction metadata, such as the relevant smart contract name and version; the signature of the transaction initiator utilised to verify the transaction and to ensure the transaction is not tampered with; the proposal, which sees the input parameters sent to the smart contract by client application; the response, which is the result of the read and write set values of the smart contract; and the endorsement, which is the signed node response of transaction that satisfies the endorsement policy and refers to a list of endorsement responses for each transaction proposal.
- **Metadata:** This contains information about the block not included in the block header, such as how to create the block, creation timestamp, the block size, the public key and signature of the creator. In addition, it indicates if the transaction was valid or invalid and had a random value of nonce, also called a counter, which is crucial to solving mining if the blockchain uses a mining process; otherwise, may this value not be included.

2.3.2.2. Hash Functions

The hash function is a cryptogram algorithm widely used in numerous areas, such as the financial industry, communication, data recognition and so forth. The algorithm is a one-way function that encrypts an input text with any dynamic length to produce the hash fixed-length output [79], [81]. The hash function has been involved in numerous applications related to data and system securities, such as password checking, authentication protocols and digital signatures, to provide protection. It has numerous features [82]:

- **One-way:** It is easy to compute and obtain the input from the hash output.
- **Deterministic:** This means that processing the same inputs produces the same hash outputs.
- **Collision-resistance:** If one bit in the input is modified the hash output is also changed. Therefore, different inputs lead to different outputs.

There are various types of the secure hash algorithms (SHA) family. In 1993, SHA-0 was the first SHA family developed by the US National Institute of Standards and Technology (NIST) [83]. SHA-1 [84] was then published in 1995 after discovering a significant weakness in SHA-0. Both hashes encrypt input text up to 264-1 bits into 160 bits. Both also use 32 bits for word size and 512 bits for block size. However, SHA-1 computes hashing with extra steps that

address the SHA-0 problems. SHA-1 is applied in common security tools and Internet protocols, such as SSL, SSH, S/MIME, TLS, IPsec and PGP [85]. In addition, it is used in the private sector as a protection scheme to hash sensitive information as well as being used for authentication and integrity realisation models [86].

Between 2001 and 2004, SHA-2 variants were published by the NIST and demonstrated more robustness than the previous ones. The SHA-2 family includes four algorithms: SHA-224, SHA-256, SHA-384 and SHA-512. The former two algorithms accept the same block size as SHA-1 for input messages, which are 512 bits. However, they produce different output sizes: SHA-224 creates 224 bits, and SHA-256 generates 256 bits. The latter processes input messages up to 2128 -1 bits, and they increase the block size and the word size respectively by 1024 bits and 64 bits. SHA-384 produces 384 output bits, while SHA-512 produces 512 output bits. In 2012, a public competition was opened by NIST to develop another hash algorithm. Keccak [87], the winner of this competition, developed SHA-3, which was chosen as the next secure one. SHA-3 has the same SHA-2 length, although its internal structure is different and more complicated than the previous SHAs to ensure security and enhance computation efficiency. SHA-2 and SHA-3 hash functions are both currently in use.

Secure hashing algorithms are utilised in blockchain technology - commonly adopted is SHA-256 from the SHA-2 family [88] - to secure the networked system, specifically the identities and data shared in the network. Many papers have developed their own hashing algorithms or enhanced the existing secure hashing algorithms. However, both approaches are out of this thesis' scope since the thesis analyses and uses the current blockchain platforms with their adopted secure hashing algorithms.

2.3.2.3. Consensus Mechanisms

In a distributed environment with no centralised authority, consensus mechanisms are required to ensure a dependable and trustable system. The consensus mechanism is a protocol that allows all mining peers responsible for appending valid transactions to the ledger to receive all submitted transactions and reach an agreement in the network. This protocol is considered the backbone of blockchain technology. There are numerous consensus algorithms that apply to various platforms of blockchain networks to ensure the consistency and integrity of stored data across geographically distributed peers, such as proof of work [89], Byzantine fault tolerance [90], proof of authority [91], and poof of stake [91]. Chapter 4 will discuss consensus mechanisms in more detail, including their types, functionalities and characteristics.

2.3.2.4. Transaction

In a business environment, a transaction is used to describe a transfer from the seller to the buyer. In the blockchain, the transfer is documented in a ledger [92]. A transaction can describe any interaction among peers in the blockchain. For example, in the bitcoin blockchain, a transaction refers to the cryptocurrency transferred between peers in the network. The transaction's life cycle begins when the user initiates and digitally signs it [93]. The digital signature employs the user's private key during the generation process since each participant in the blockchain has a private and public key, as shown in Figure 2-2.

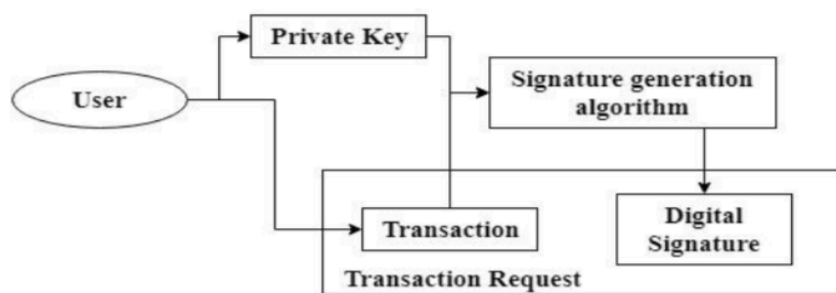


Figure 2-2 The process of signing the transaction [93]

The signature generation algorithm uses the sender's private key and the transaction value to create the user's digital signature. The signature is then attached with the user transaction to be broadcast to other peers in the network in a phase known as a transaction request. Then, the transaction request needs to be vitrified and validated by the miner. The miner uses the sender's public key in the validation process to add the transaction to the blockchain. When the network has more than one miner that works simultaneously, a consensus algorithm is required. When the transaction is added to the chain, it cannot be altered, although it can be reviewed.

2.3.2.5. Ledger

A ledger can be called for a list of transactions. In the past, old-style ledgers were registered using paper and pen to track transactions, such as exchanging services and goods. Ledgers then modernised by using digital databases which rely on a trusted centralised third party who owned the ledgers: centralised architecture leads to numerous issues, such as a single point of failure, distributed denial of service (DDoS) and denial of service (DoS) attacks, that are explained in more detailed in the next chapter. Recently, blockchain technology has allowed the distribution of ledgers instead of doing so in a centralised fashion. There is growing interest, and much research has been conducted to develop distributed ledgers to increase security, reliability and trust. Therefore, a distributed ledger is a replicated copy of a ledger distributed

among nodes on the network[92]. In other words, data or assets in the distributed ledger can be recorded, shared and synced among peers that reach consensus. Thus, the information can be secured in the future, and all participant nodes are responsible for maintaining the ledger.

2.3.2.6. Smart Contract

In 1994, Nick Szabi [94] introduced the concept of a smart contract as a transaction protocol that can be computerised and run the contract terms. Later, he reused one of the primary blockchain components as a set of code or programming instructions that can be run by the participants in the blockchain network to handle the transferred data between peers and to ensure its consistent updates. This script works as an agreement between parties that requires independent execution and agreement on the terms of the contract backend before adding any transaction in the blockchain to increase trustworthiness, minimise malicious occurrences and eliminate human judgement. It also consists of several protocols that combine with user interfaces to secure and formalise relations in the network. Each peer has its own rules stored in smart contracts to negotiate the contract terms and execute the agreement. In addition, smart contracts have principles and objectives to design the terms derived from secure protocols, economic theory, legal principles and theories of reliability. Each smart contract has its unique address since it is stored on the blockchain, and any executed transaction stores this trackable address that cannot be altered.

Originally, the smart contract was used in economic and financial services within a limited range of transactions. Recently, it has been deployed in a broader range and converted different contracts from different fields, such as education, supply chain and healthcare, into programmable scripts [95]. In addition, smart contracts can be either fixed or can be self-programmed depending on the blockchain platform. Some platforms use different calls for smart contracts, including chaincode, as in Hyperledger Fabric, which will be addressed in more detail in Section 5.2.1.

2.3.3. Blockchain Background and Generations

In 2019, Casino et al. [96] classified blockchain technology into three generations: Blockchain 1.0, which started when Bitcoin was founded with all cryptocurrency transaction applications; Blockchain 2.0, which extended beyond digital cryptocurrency applications and included all applications with smart contracts; and Blockchain 3.0, which used blockchain applications in a broader context, such as healthcare, government, smart city and so forth. Blockchain technology is a trend that is growing quickly, and in 2020, the blockchain developed into its

fourth generation [97]. Blockchain 4.0 is related to Industry 4.0, which supports all decentralised Blockchain 3.0 applications. The two can also communicate with each other.

When this thesis was started, it was designed for Blockchain 3.0, then the most-current blockchain generation. By applying blockchain to decentralised current smart campus frameworks, the study was extended to store data on-chain and off-chain to increase the scalability, which is considered one of the 4th generation aims, and integrate into Industry 4.0. Both generations are still in their early stages, and experiences are maturing. The following sections briefly present the evolution of blockchain technology generations.

2.3.3.1. Blockchain 1.0

The vital blockchain's technical elements, such as smart contract, byzantine fault tolerance and public key infrastructure, are not recent developments, and they have been identified individually since the 1980s and 1990s [72]. In 2008, Satoshi and Nakamoto [98] combined these components together into the blockchain system. They wrote a paper introducing a bitcoin and electronic cash that allows peers to send transactions directly without needing to go through central financial institutions or any third party. In that research, Nakamoto invented the ledger, which was named 'a chain of blocks'. This chain of blocks involved developing a new version of cyber currency and electronic cash [99] and was later called blockchain in 2016 [100]. In this generation, the idea was primarily designed to create digital cryptocurrencies and payment functionalities. However, the primary disadvantage was that transactions needed to be controlled. Since Nakamoto initially introduced blockchain technology, numerous other blockchain technologies have been studied and improved.

2.3.3.2. Blockchain 2.0

From 2010 to 2013, blockchain became commonly used in cryptocurrency applications, digital payments and currency transfers. The Ethereum platform [101] was proposed in the second level of blockchain, introducing the idea of the digital ledger or the smart contract, a self-executed programme or computer protocol that works as an automatic agreement between nodes. A smart contract comprises all the conditions and clauses involved in financial services and applications and is transparent to all peers. Scalability and sustainability are the significant downsides for this generation.

2.3.3.3. Blockchain 3.0

Blockchain changed the finance industry services, but it also has influenced other areas, such as business [102], the health sector [103], and security [104]. At this level, smart contracts have been improved and can shift the blockchain towards a decentralised internet, which means integrating smart contracts, communication networks, data storage and open standard platforms. Many platforms have been developed, such as the Hyperledger framework [105], which can decentralise various applications called DApps. DApps run blockchain networks in the backend to decentralise the applications and have a frontend user interface coding in any programming language that can call the blockchain at the backend.

2.3.3.4. Blockchain 4.0

With the growth of decentralised applications, there is a requirement to integrate several architectures and services under one umbrella platform to communicate with each other. Therefore, users from different platforms can cross-chain communicate as they work as a single unit. With Industry 4.0, there is a requirement for seamless integration of various execution platforms with an increased degree of privacy and trust. There is also a requirement for increased scalability of blockchain networks [97]. Blockchain 4.0 fits with Industry 4.0 requirements and allows the integration of business systems to operate cross-chain and ensure security. In other words, this generation of blockchain technology focuses on Blockchain 3.0 and includes distributed databases and a public ledger. This development means they can operate in real-time business and meet industry logistics to fulfil the requirements of Industry 4.0. The SEELE platform [106] is an example of this generation and allows individual blockchain systems to connect to each other while they operate independently. In addition, it supports linear scalability by operating a neural consensus algorithm and computes on-chain and off-chain.

2.4. Blockchain in Education Environments

The increased number of studies and developments of blockchain in recent years indicates that researchers have involved it in many different areas beyond finance and currency. Due to its properties of immutability, decentralisation, reliability and data provenance, it is a suitable technology for developing in various fields, such as smart education systems [107], [108]. Many studies have discussed its benefits to education systems and applications, including

administration work, fee payment, library managements, academic records, certificates and publishing.

This section presents blockchain-based education projects and classifies them based on their services to help understand how blockchain technology assists in this area and what kind of blockchain platforms and consensus algorithms have been used to develop the smart education framework that will undergo discussion in Chapter 4.

2.4.1. Blockchain-based educational institutions

Numerous educational institutions and universities deploy blockchain technology in their applications to support administration work, maintaining certificates and fee payment. For example, the University of Nicosia, Cyprus [109], [110] is considered the first university to use the blockchain to issue certificates and accept bitcoins for tuition fees. The university developed the Massive Open Online Course (MOOC) platform to teach its course on cryptocurrency and offered an MSc. degree in Digital Currency. The certificates are issued using PoW consensus algorithms and use the SHA-256 algorithm to generate an appended unique code to share without amends. The Maltese Education Institution developed a blockchain platform to store the credentials of its employees and learners. The platform uses the Hyperledger blockchain and PoW consensus algorithms to achieve the institution's objectives of allowing its users to manage and control their own data and share their credentials globally. In 2017, the Massachusetts Institute of Technology (MIT) [109] started to issue and verify digital certificates to their students and give them diplomas using the Blockcerts platform, which is based on the Bitcoin blockchain, with the support of a learning machine. The University of Birmingham [111] developed a blockchain platform named BTCert for issuing certificates based on the Bitcoin blockchain. BTCert resolves the issues related to Blockcerts, which have limitations related to authentication, by implementing a set of novel cryptographic protocols. Since 2015, Open-Source University [111] has offered a platform for academia, students and businesses. The platform uses the Ethereum blockchain and IPFS to authenticate the learners' credentials and allows other employers and educational institutions to verify learners' certificates. In addition, the platform uses a smart contract to support payment options.

2.4.2. Blockchain-Based Learning Platforms

Many researchers have used blockchain technology in their learning platforms. For example, from 2018, Woolf University [112] was an early blockchain university designed to be a decentralised institution and increase transparency between lecturers and students. The university uses Ethereum and smart contracts to teach courses internationally and rewards both learners and academia. Dubai has launched the largest blockchain model for learning platform named Educhain [111], based on Ethereum and smart contracts to allow more than 400,000 learners to use the platform. EduCTX [113] is another project developed for the University of Maribor using the Ethereum blockchain and a Delegated Proof of Stake (DPoS) consensus algorithm to allow a higher educational institution to have a right to access the network and offer benefits for learners and academia. The decentralised platform allows students to verify the courses and means academia can issue and verify candidates' certificates without a third party. In the application process, EduCTX allows institutions to validate applications and provide a digital certificate to their learners who attend workshops, lectures and training. Light Chain [114] is a blockchain platform for online quizzes. The framework evaluates the results and stores them on the blockchain to provide immutability for manipulation. SuccessLife [111] is a platform for organising global workshops and seminars. It contains hundreds of workshops and seminars across 30 countries and has over ten million participants.

2.4.3. Blockchain-Based Library Management System

Library played an important role in intellectual developments and social enlightens. Many researchers have been focused to emerge new technologies to develop smart knowledge and shared services to the library systems whether they are public libraries or university libraries. Most of the current library systems are rely on third party and centralised architecture. The common services that library management systems provide are circulation, cataloging, reference, journals, and public consultation. These modules mostly shared central back-end databases to ensure control on their data [115]. However, these library services could face many practical issues, such as fast transmission speed, complex interaction types, huge amount of data and ensuring users' privacy. In addition, the centralised structure of these institutions leads to be self-administrated and remained isolated which cause to less or poor usage of their resources. Therefore, with the features of blockchain technology including decentralised data structure and immutability of the network, it has been emerged in the field of developing library management systems to solve these problems.

Although blockchain technology have been developed in different areas and applications, there are small number of studies in the field of blockchain based library managements and services [116], [117]. For example, Liu [115] used the advantages of blockchain technology to design a book management system and discussed theoretical the borrowing a book process function including entering the borrower information, generating the book borrowing, and transferring and returning the book. Cabello et al. presented a project idea named LibChain which blockchain technology was exploited to decentralise current library systems and eliminate the bureaucratic obstacles of library services for both providers and users. However, these schemes were discussed the idea without presenting implementation and experimental results. In addition, Zeng et al. [118] developed book sharing system named BookChain. It used BCOS blockchain platform to enable students to borrow and returned their spare books. The system allows proper traces and supervision of each book in a campus without intermediate such as libraries since the book information and status stored in the ledger. Chiu et al. [117] proposed library system called LibBlock which aimed to provide decentralised, robust, adoptive and flexible e-library. The system emerged Ethereum blockchain technology and IPFS to enrich the efficiency of the current systems.

2.4.4. Blockchain-Based Academic Records and Certificates

Blockchain has been employed in the field of storing academic records and issuing and verifying educational certificates due to its characteristics of immutability and decentralisation. With blockchain technology, any respective educational institutes can update the ledger of all students' achievements, grades and certificates. In the future, these documents will be easy to verify by checking the records ledgers. For example, Sony Corporation and Sony Global Education developed an education system, built on the IBM blockchain, to store and share students' records among permissioned peers [119]. APPII [111] is a platform using the Ethereum blockchain to verify the academic qualifications and backgrounds of teachers and learners. APPII, in partnership with the Open University, uses a smart contract to verify users' career history documents. In addition, Gradbase [120] is a platform for storing and verifying educational records and QR codes attached to students' documents. The platform is based on the Bitcoin blockchain and PoW consensus algorithm to allow users to update and delete their professional and academic qualifications. The platform has a tie-up with LinkedIn for an additional feature. Stampery [111], [121] is another scheme to verify data using Bitcoin and the Ethereum blockchain. Binding both blockchains' consensus algorithms with routing keys increases efficiency and scalability. TuringChain [122] is a project to track records to ensure

sustainability and is unified for the education field instead of traditional educational certificates. It uses the Ethereum blockchain with AI to rely on the Turing test. Edgecoin [111], based on the Stellar blockchain and PoS consensus protocol, is used to store employment history, educational credentials and candidates' skills. It uses smart contracts to automate recruitment processes. RecordsKeeper [110] is a system for issuing e-certificates and generating a receipt for an educational institution. The receipt number, along with key pairs, will be used to verify the certificates. RecordsKeeper was developed using the Multichain blockchain, which is a fork of the Bitcoin blockchain. DISCIPLINA [111] is an educational platform based on the Ethereum blockchain and the DPoS consensus algorithm that aims to help recruitment companies and academic institutions store candidates' achievements and backgrounds. The platform developed a scoring system to allow institutions and employers to search for a potential candidate. Aastha et al. [123] developed a framework based on the Ethereum blockchain and IPFS to issue a digital certificate and verify it using student identities.

2.4.5. Blockchain-Based Publishing

The field of scientific publication is another area that can leverage blockchain technology. The main issue in this field is the system for submitting manuscripts. This is because, in most cases, the system provider is a third party that supports organisers of journals in terms of coordination and academic conferences in terms of the submission workflow, which starts from submitting the abstract and paper and moves onto arranging peer reviews and camera-ready submission. Although such systems have made the manuscript submission process more efficient, they have also raised concerns about potential dishonesty and technical weaknesses. For example, the naming scheme for all paper submissions made to Sheridan Printings, which is a software for conference management utilised by several conferences (e.g., ITS and TEI), was easily guessable for the 2004-2011 period. In particular, the naming scheme made it possible to retrieve all the documents submitted to a particular conference easily for anyone with the base URL [124]. Thus, in such cases, a researcher must trust the system when submitting their unpublished work, which leads to the possibility of data leakage or idea leakage. In addition, users must trust that anonymous peer reviewers and the program committee will not plagiarise or amend the results or findings.

Many research papers have proposed the use of blockchain systems to resolve these issues and to decentralise manuscript submission systems. For example, Gipp et al. [124] developed a scientific manuscript system named CryptSubmit to provide data integrity and security for

submissions, which relied on the use of Bitcoin blockchain technology to verify timestamps as a tamper proof data point connected to each submission. Hepp et al. [125] proposed OriginStamp using Bitcoin blockchain technology for intellectual property protection, offering a novel method for timestamping and storing documents. Pozi et al. [126] proposed a framework based on blockchain technology that used PoW to measure writer contributions and to enable the paper to be accessed transparently. In addition, Andi et al. [127] used blockchain to design a plagiarism prevention model, along with digital signatures to protect the work and allow only reviews to access the paper without altering it.

All the above projects and schemes use blockchain technology for their educational applications in different contexts and have varying aims, such as increasing their systems' security, privacy, data integrity and data protection, enhancing transparency and responsibility, increasing trust and facilitating communication among parties. However, these studies have limitations that can be summarised as the following:

- The studies used different blockchain platforms, mostly Bitcoin and Ethereum blockchains, with different consensus algorithms which lead us to conduct research, as will be discussed in Chapter 4, to help select and develop a suitable blockchain platform for experimenting with applications in educational environments, such as students' transcripts, certificates, credentials or any other accomplishment data forms.
- Most of the above studies had not addressed the scalability issue when the data size becomes too large to fit on the blockchain, which leads to increased transaction latency. In Chapter 5, the solution of this issue will be covered by storing most of the data off-chain using IPFS.
- There are a few pieces of research, such as Open-Source University, that use off-chain storage. However, they developed them with the Ethereum blockchain, which is, according to this study, not a suitable type of blockchain to use in an educational environment, as will be discussed in more detail in Chapter 4.

2.5. Summary

This chapter presented an overview of the smart campus concept with a discussion of the primary technologies and platforms deployed in it. It studied eight domains in the smart campus and defined problem assets per domain. Then, the chapter presented an overview of blockchain

technology and its components. All blockchain generations had been presented with their fundamentals to help categorisation in this thesis and to contribute to this field. Blockchain technology has grabbed the attention of academic and educational areas. By joining distributed ledgers, hashing mechanisms and cryptographic philosophy, data can be exchanged in an immutable and clear way. The chapter highlighted the benefit of the technology in this field and demonstrated the limitations of recent studies. The next chapter will propose the novel smart campus architectural framework.

Chapter 3 Analysis of Smart Campus Architectural Frameworks

Rapid development in advanced technologies, such as the internet of things (IoT) and blockchain, has led to a growth of interest in a number of application areas, including smart environments, such as a smart campus, which can be considered as an example of a smart city. Despite such growth, no comprehensive guiding framework has been developed for emerging IoT and Blockchain technologies deployment in the smart campus environment, particularly in relation to security and privacy aspects, as well as to the mitigation of known problems with IoT and Blockchain in existing applications. This chapter helps to answer the second research question: ‘How can the smart campus technologies be characterised?’ to propose a novel architectural framework for the IoT and Blockchain applications deployed within a smart campus environment by comparing the main technologies involved. Then, the framework will be discussed, with particular consideration to security and privacy requirements.

3.1. Overview of Current Smart Campus Frameworks

Many researchers have proposed different smart campus frameworks that would need to be reviewed and studied in order to develop the suitable smart campus framework. Most of these frameworks have been used IoT architecture to propose smart campus environment. The middleware layers -sitting between the hardware and sensors layer and the applications layer- in IoT architecture have almost the same functionalities; however, the underlying technologies are different. Thus, it is important to address these frameworks based on IoT architecture layers.

Some researchers have been developed a smart campus framework using the traditional three-tier structure. For example, Narendrakumar and Pillai [128] proposed the *Smart Connected Campus* framework to deploy and exploit IoT technologies in a campus setting in order to

remotely monitor various campus activities. The architecture generates several features reliably and easily at run-time, such as water and temperature monitoring systems, a route map and online resources, and integrates them all in a single platform. All the facilities are communicated and networked through IoT, enabling data generation at run-time and remote access to the data. The framework contains three major components: sensor technologies, a cloud server, and Android mobile applications. In addition, authorised login credentials are used to authenticate data access. Furthermore, Hossain et al. [129] proposed a smart campus model based on IoT technology, which included campus facilities, such as smart parking, smart classroom, smart buildings and cloud computing system. The model consists of three layers: a perception layer which is responsible for sensing data from IoT campus devices; a network layer which sends captured data to cloud storage and stores collected data via Wi-Fi or the internet; and lastly, an application layer, which provides services and applications to the end user.

Zhe et al. [130] developed a smart campus-based information service. The framework integrates various service information systems with the individual's location to provide unified information service on campus. The architecture uses the basic three-tier structure. The presentation layer, mainly based on mobile applications to present the information, includes messages, location and statistics to users. The application layer comprises location applications and local information services. The data layer consists of three components: data source, including location data and different applications data; basic data, which is utilised to store basic and location information; and data interface, which supports the interaction between the system data and the location service platform. The study did not, however, mention blockchain technology.

Moreover, Yan and Hu [131] constructed a smart campus framework-based data platform, providing the association with data analysis using Apriori algorithm as association mining algorithm. The framework consists of basic three-tier structure: the user interface, which shows the results of data mining; the middle layer, where data is processed; and the data layer where the data is stored in a database. In addition, Hu and Yan [132] proposed a smart campus-based big data framework to offer cooperation and interconnection between different applications systems. In addition, the framework used cloud computing as big data storage and the K-Means algorithm to process the data. However, all of these smart campuses used traditional databases, which suffer from third party issues that will be discussed in more detail in the next section, and blockchain could be the solution to storing the data. The frameworks could use blockchain technology to increase the security and protect the data.

Some studies have been developed smart campus frameworks using four IoT architecture layers. For instance, Agate et al. [133] designed a smart campus-based fog computing framework to enable data collection from various smart devices. The edge of the network enhanced the services available to users and improved the user experience in the campus. This framework based IoT architecture consists of four layers. The lowest layer of the framework includes various heterogeneous sensors, which are responsible for collecting raw data and sharing it with the higher layer. The next layer consists of fog entities, which received data transmission from edge devices using communication infrastructures, such as Wi-Fi, aggregating all data and sharing it with the cloud. The cloud layer computes and analyses the data, as well as being responsible for storing all information needed to display the collected data and results on the application layer. This approach focuses on integration of fog computing and IoT to improve the scalability of the system. However, such a system usually includes security and privacy issues.

Enqing et al. [134] constructed a smart campus and three dimensional geographic-based platform to achieve data and service sharing, as well as the management of heterogeneous and multi-source data, such as spatial real data, information database and geographic information. The framework contains of four layers. The first layer is the infrastructure layer, which includes hardware, network devices and the operating system. Then the data service layer has different databases, such as a spatial real database, geographic information and other databases. Next, the platform function layer includes a basic function to support the 3D geographic information interface and the business function, which supports data interface, such as image and video queries. Lastly, the application layer is responsible for providing application and data services. This study did not cover blockchain technology.

Lihong [135] constructed a smart campus based on IoT and deployed on Hadoop as cloud server. The framework provided resource sharing, dynamic understanding and information released real-time. The architecture was divided into four layers: a data layer, responsible for storing and classifying data based on business requirements and data characteristics; a system layer, comprising operating system and several subsystems, a central server hardware platform and the layer supports that connect other layers; a network layer, which includes the IoT networks that support different types of sensors for coverage and transmission of various resources and information on campus; and an application layer presents obtained data to users and provides interactive services through management processes and business rules. However, the framework did not study the integration of blockchain technology into its architecture.

A five-layered IoT has also have been used to develop smart campus frameworks. For example, Agarwal et al. [136] proposed a five-layered IoT smart campus framework that can be implemented in various cases. The first of the five layers is a sensor and data acquisition layer, which is responsible for capturing data in real time from different sensors then sending the data to the upper layers. A compute and infrastructure layer comes next, which consists of the required infrastructures, such as Wi-fi, Zigbee and BLE that are needed for network connectivity by gathering data from the previous layer and send it to the cloud or backend server. Then the platform layer is responsible for developing a unified layer to support communication among heterogeneous systems, integrate the data from various systems and define business rules. The application layer helps by defining and implementing different usage scenarios within the smart campus. Finally, a monitoring layer monitors the applications and creates alerts when needed. A dashboard and command centre can be set up to monitor applications on a regular basis. The framework of smart campus can help to implement different cases based on usage; however, it relies on IoT technologies and third parties to store the data, which leads to several issues that will be discussed in more detail in the next section.

Lastly, some researchers have proposed a smart campus framework using seven IoT architecture layers. For example, Debauche at al. [137] presented (R)evoCampus as a smart campus architecture based IoT technologies. The seven layers consists of: infrastructure layer responsible for collecting environmental information or understanding an action; an information layer allows different objects to identify and communicate with connected devices; a communication layer, responsible for data transmission among connected devices using various technologies and protocols, such as Wi-Fi, 4G, and Zigbee; a connective layer, allowing interoperability and connectivity of exchange data among devices; a middleware layer safeguards the storing and processing of data in the cloud; a service layer delivers reliable and critical facilities for several applications, such as open data and weather station; and finally, an application layer provides various data consultations, visualisations and presentations to end-users. However, the framework did not mention blockchain technology.

In conclusion, all of the studies mentioned above have proposed smart campus frameworks based on IoT technologies. Each framework has been developed using different IoT architecture middleware layers, which mainly sit between the hardware and sensors layer and the applications layer. The middleware layers contain a set of functionalities, such as data storage, data management, and data processing, which are required when developing smart applications, since data is the heart of smart applications and IoT environments. Therefore,

data has to be secured and must not be stored in third parties. This will be discussed in the next section in order to propose the smart campus framework, which stores data in a secure way, as well as using suitable IoT middleware layers, providing appropriate technologies and tools in each layer.

3.2. Existing Problems for Current Smart Campus Frameworks

Although the above existing smart campus frameworks provide different architectures with a different number of layers, they all rely on an IoT centralised architecture, which is developed using a centralised server, to manage and control connected nodes. In other words, a centralised architecture is a client-server architecture, that typically has a centralised server cloud computing and clients that are represented by IoT nodes. The centralised server has many roles, such as processing data, managing task scheduling, dealing with all requests coming from the network and storing information.

The centralised architecture provides several advantages, it allows a variety of devices to connect and communicate among each other under the management of the centralised unit and provides the needed identification and authentication for connected devices. The whole network, in this case, is easy to control and is maintained through central server. In addition, the architecture does not need to be installed in many workstations requiring software and hardware since most of the operations are done by the centralised unit, which also saves on costs.

However, the centralised architecture suffers from various threats and complications. Khan [138] and Atlam and Wills [139] discussed these issues in details as summarised in the following:

- **Security:** Many studies have been done and have confirmed that the most serious issue for the centralised system is security [140]. Since all the system operations are executed via a central unit and all data is saved in one place it increases the risks of being a target for different types of attacks. Denial of Service (DoS) and distributed denial of service (DDoS) attacks are most common information leakage attacks. Therefore, there is a high risk in storing sensitive data in a centralised server.

- **Privacy:** Another vital issue that needs to be considered is privacy, since smart devices collect a massive quantity of confidential and personal information such as passwords, financial accounts etc., and is stored in centralised unit which usually belongs to third party and can easily be attacked. There are some examples that service providers attack the privacy of users [141], [142]. For instance, some third parties such as Facebook sell user information to marketing companies, which in turn use this information to analyse user behaviour in the network [142]. In addition, data can be altered or deleted from insider attacks that will affect data integrity. Thus, a secure method is necessary to provide privacy of information.
- **Single point of failure:** When all operations, controls and storage is achieved by a centralised server, it can create a single point of failure which means when the central unit is down, the whole system will fail and will be unavailable [141]. The common way to avoid this issue is by adding redundant switches, servers, and network connections as a backup to work alternately when the original centralised server fails. Nevertheless, this approach has many issues, such as it is expensive to install the alternative requirements, and that there are synchronisation problems between the backup and the original server.
- **Scalability:** Another main issue for a centralised architecture is scalability [141]. The centralised unit uses a central authority to process all commands and controls which can be scaled for small networks however, it can be impractical for large organisations such as campuses, that are distributed in different areas, which would increase the number of IoT devices needed. According to Piekarska and Halpin [143], there are concerns about the efficiency of operating and the scale of the IoT system with centralised architecture taking into account the increasing demands. In this case, transporting decisions for large organisations may suffer and not operate efficiently.

Recently, blockchain technology has been involved in various application areas beyond the cryptocurrency domain since it has multiple features, such as decentralisation, support for integrity, resiliency, autonomous control, and anonymity [144]. In addition, blockchain shows several properties lead it to be a suitable technology to apply it for different applications as it is discussed in the following:

- One of vital properties is that blockchain eliminates a central authority by using a distributed ledger that allow to achieve a distributed consensus in order to build a

decentralised system to provide more efficiency for operating and controlling communication among all participating nodes.

- Blockchain ensures immutability since an accepted transaction stored in distributed ledger in a block then each block have to reference its previous block to build a chain, therefore, any published block did not refer to its preceding block, it would be rejected by other nodes.
- blockchain ensure data persistence since it is stored in a distributed way which means many copies of the same ledger shared, updated and synced among nodes. Even though if a new peer joins the blockchain system, it can reach of a full copy of the distributed ledge, therefore, the ledger is difficult to be lost or destroyed. Unlike cartelised systems who owned the ledger; they should provide data persistence by backing up the system and a user have no choice to trust them.
- Data in blockchain is provenanced when the transaction or data stored in a distributed ledger, it needs to be processed through cryptographic mechanisms, such as hash functions and digital signatures. This process ensure tamper resistant and authenticity of the transactions. While on centralised system, a user has to trust the owner of the ledger and transactions not being altered.
- Blockchain eliminates the single point of failure since data stored in and retrieved from the distributed ledger. While in the centralised platform if it goes down, which could lead to the failure of a whole system [139].
- Another property of blockchain is transparency since all transactions are valid before adding to the blocks. If there was an invalid transaction made by a malicious peer, other nodes would detect and reject the transaction. Unlike a centralised model, a peer has to trust that received transaction is validation by the owner of the ledger.

All these properties put it in the lead as a suitably advanced technology to apply to the smart campus framework. Therefore, deploying blockchain technology to the proposed smart campus framework will provide more benefits by managing the problems associated with a centralised IoT architecture, especially from a security perspective.

3.3. Proposed a Smart Campus Architectural Framework

A novel smart campus framework architecture based on IoT and Blockchain has been designed to provide a comprehensive guiding framework [107]. The main goal will be the collection and aggregation of the data from various areas of the campus while increasing the data security and providing a better service to enhance the experience of the user. To accommodate the different areas of the campus, several devices and sensors will be used. These will be distributed around the campus and will work simultaneously with an aggregation of networks and software to generate heterogenous data, which will be released as reliable and valuable data from the IoT. Thus, there is a need for the system to support several network protocols to provide scalable and reliable communication across the networks. In addition, the system needs to handle the previously discussed problems related to current smart campus frameworks, especially security.

IoT architecture with six layers is applied for this framework. It employs Blockchain technology as a cross-layer component to eliminate the third party, decentralise the architecture and ensure data security. See figure 3.2. for the conceptual framework.

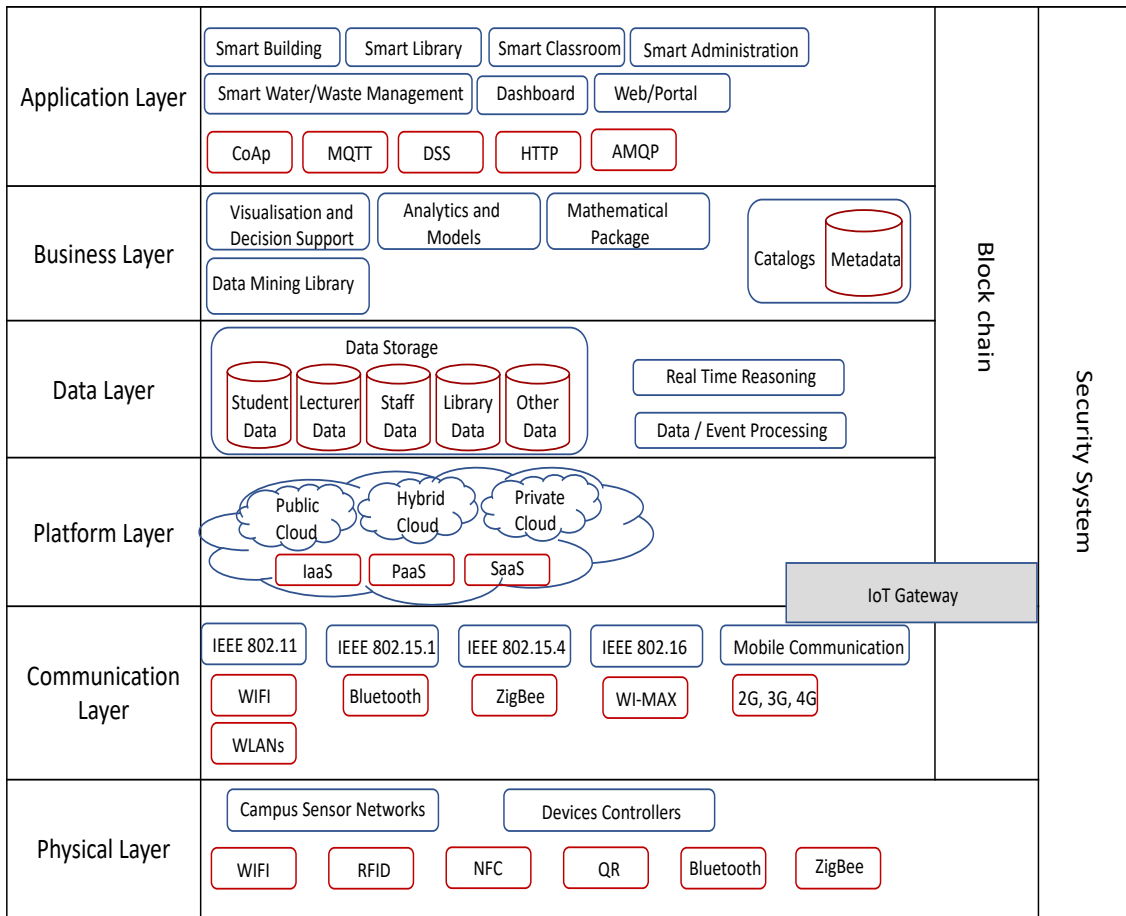


Figure 3-1 A novel smart campus framework [107]

The following subsections describe each layer in more detail:

1 Physical Layer:

On the lowest level of the proposed architectural framework, the physical layer, several sensors have been used to collect and share raw data with upper levels. There are two types of sensors in the system: deployed and wearable. The former is at fixed points in various areas of the campus such as RFID and GPS. They can be installed at these locations to gather environmental data, such as humidity, temperature and location. The wearable type of sensor can be used by individuals daily in items such as tablets and smartphones. When the sensors sense the physical campus environment the parameters are then converted to data signals to be handled on the Cloud for analysing. The layer also contains actuators operate in the opposite way: that are responsible for the physical action of converting data signals coming from the upper layers [145] perhaps as a response to sensor data stored on the blockchain, which is subsequently analysed and results in an actuator event. Thus, smart devices are defined as any hardware component that can be connected to the internet and can operate with or without wires.

2 Communication Layer:

The communication layer is sometimes known as the network layer or transmission layer [146], [147]. Since IoT is about interoperability and connectivity, there is a need for communication between objects. The different data sources that are provided by the perception layer need to be connected to the upper architecture layers to handle collected data. Devices and sensors use protocols and adequate communication technology to connect to the Internet. An application such as a smart campus has a variety of data sources, which require the use of diverse communication technologies. For example, Wi-Fi/IEEE 802.11 utilises radio waves to allow smart devices to exchange and communicate within a 100 m range with power efficiency and without utilising a router in some ad hoc configurations [63]. IEEE 802.15.1 standard uses short-wavelength radio to exchange data between smart devices and to minimise power, such as Bluetooth low energy (BLE), which operates for a longer period of time and within a 100 m range. Recently, BLE was considered a suitable technology to support IoT applications [148]. In addition, IEEE 802.15.4 is another communication technology protocol ideally suited for low high-message throughput, low cost, low data rate, and low power consumption and is also a good candidate for machine-to-machine (M2M), wireless sensor network, and IoT. This standard is used to produce Zigbee protocol for more reliable communication and a high level of security [63]. It can be used in applications that need low battery life and a low data rate. Cellular or mobile communication technology operates over long distances with high throughput data, such as 3G/4G/5G [149], and allows devices to connect to the Internet and communicate at a reliable high speed. This is a good technology to use for mobile devices applications. Thus, the communication level is responsible for broadcasting the data across the layers in the network, the initial connection setup and dealing with data transmission errors. It connects the various communication protocols and sensors to the internet via IoT gateways.

The IoT gateway acts as a bridge for various smart devices within the internet or public communication networks [150]. When different sensing domain protocols communicate with each other or when sensing domain protocols communicate with a network domain protocol, the IoT gateway supports protocol conversion to enable them to communicate. In addition, it can identify, maintain, diagnose, configure and control smart devices. In other words, the IoT gateway for different networks acts as a proxy and solves the integration issues.

Many have studied integrated Blockchain technology in this layer to provide security in communication. Biswas and Muthukkumarasamy [151] proposed a smart city framework that

integrated smart devices with blockchain technology to support secure communication. The paper stated that the Blockchain needed to be integrated with the communication level, converting the transactions into blocks using Telehash to be sent to the network. This would provide privacy and security.

3 Platform Layer:

Generally, a smart environment based on IoT uses a large number of data sources, including actuators and sensors that produce big data, which need to extract knowledge by using complex computations, applying data mining algorithms, and managing the services and allocation tasks [152]. Traditional information systems are not effective for this as they are prone to data inconsistency and have no data sharing capacity [153]. Therefore, there is a need for advanced technology in the processing unit that can provide the resources needed and have the computational capability. Thus, cloud computing presents the suitable technology and a powerful computational resource for IoT to process, compute, and provide valuable results. Blockchain can be integrated with this layer to decentralise the system structure and can be used instead of cloud storage to store the data.

4 Data Layer:

Campus systems have a massive amount of data called 'big data' that needs to be placed in a data layer. The layer can store and retrieve data and contains all the data that the campus needs in a variety of forms. Instead of storing campus data over the network, Blockchain technology can be used to decentralise data storage and use a data warehouse to add privacy and security.

5 Business Layer:

Every organisation has specific objectives and goals that need to be accomplished. For this, they need to collect intelligence from analysed data and apply it to their business strategy and planning. This layer delivers an evaluation of the performance of existing services and applications and provides intelligence solutions. The success of an IoT system not only depends on the use of advanced technology but also depends on who the system delivers the services to, i.e. the users. The business layer is responsible for creating graphs, models and flowcharts based on analysed data, as well as creating business rules, process management and the process of decision making.

6 Application Layer:

This layer can consist of many different application types and services required by many different end users. It includes campus smart services such as learning, library, administration, building and parking. The analytical data is collected, and the information is presented in a visual form. Therefore, the application layer's main objectives are to provide high-quality intelligent services to stockholders [154], [155] and allow users to interact with the system and visualise the data via an interface.

In addition, it contains a set of protocols that are responsible for transmitting messages from the application layer. For example, Constrained Application Protocol (CoAP) is one-to-one communication protocol that is inspired by Hypertext Transfer Protocol (HTTP). It is a lightweight protocol, which is appropriate for deployment for machine-to-machine applications, particularly for communication in a local network and web services as it follows a request/response model which supports broadcast and multicast. CoAP is designed to limit fragmented messages by keeping the overhead small and causing as little traffic as possible [145]. Message Queue Telemetry Transport (MQTT) is also a lightweight protocol. It is used in a publish-subscribe-model [63]; therefore, it is appropriate for cloud/remote communication and for applications that involve short-range mobile communication, satellite links and dial-up connections. MQTT is widely used for battery-run devices as it is efficient, has a compact size, consumes less power and delivers information to multiple receivers at the same time. [145]. Thus, this protocol is able to provide an ideal messaging protocol for M2M and IoT communications due to its low bandwidth networks, low power, and low cost. Advanced Message Queuing Protocol (AMQP) is an open standard protocol that supports business applications and to transmit business messages on different platforms and focuses on a message-oriented environment. Data Distribution Service (DDS) is a publish-subscribe protocol for real-time communication [154]. Thus, merging these of communication protocols, that can each work in a different scenario and with a different device manufacturer, increases the efficiency of device communication and enables interoperation in smart applications.

7 Blockchain Cross-Layer:

Despite the benefits of IoT with connecting devices, it does have limitations in privacy and security as it relies on the centralisation of architecture, which leaves it open to attack. Blockchain technology recently merged with IoT technology. This has provided a decentralised ecosystem environment in several innovative cases beyond cryptocurrencies, such as healthcare, government and learning, with enhanced security and reduced risk. The large

volume of data in any ecosystem environment needs to be stored in such a way that it is immutable, available and accessible. For example, student data in a higher education environment is created when a student enrolls in a university. The data grows as their time at the university lengthens, and will include personal, courses, grades, finance and health information data. Student information needs to be accessible by different stockholders including administration, lecturers and finance. However, the information should be immutable and private. Therefore, a structure of storing student records should be accessible and maintainable. Blockchain allows parties to share a student record in real time with privacy, integrity and immutable features, which are all required in smart university systems. For more information about blockchain technology principles how it works see section 2.2. To decide which type of blockchain to use in the framework, the types will be addressed in a comparative analysis in chapter 4.

To sum up, a smart campus includes applications, technologies and smart features, which can be implemented in any area of the campus. There have been numerous studies done on ways to implement systems to develop a smart campus [128]–[130], however, no standard design has yet been developed. The new framework provides a guiding architecture that may allow the integration of blockchain technology with existing technologies as a base to develop various smart systems further. The framework will be analyzed in the next section using secondary sources.

3.4. The Smart Campus Architectural Framework Analysis

The main reason for developing a blockchain in 2008 was to address the potential problem related to stakeholders' trust in various use cases, including financial and non-financial fields [156], [157]. It provides security requirements for the transactions by using several cryptography mechanisms, such as signature, asymmetric cryptography, and hash. A lot of research has explored whether blockchain technology meets the need for providing more secure, trusted, and immutable data by adopting the blockchain into existing software, such as in the financial industry [158] and healthcare fields [103], [159], [160]. However, integrating blockchain technology into education institutions is still in its early stages and needs more research. Therefore, this section discusses the security requirements for the proposed framework of a smart campus since the security aspect is the main concern in most of the recent

blockchain applications. This aspect will be studied in more detail in the following subsections, covering authorisation, trust and privacy in addition to the CIA triad of confidentiality, integrity, and availability.

3.4.1. Authorisation

Authorisation is one of the key security aspects and is a process of verifying a peer's identity in order to use a system and communicate with each other [161]. There are many studies that have focused on user authentication with the majority of cases looking at data leaks and identity theft. The current authentication mechanisms, which have been used in most applications, vary from using a single factor, for example, a password or user ID, to using a multi-factor authentication, such as a smart card or biological characteristic. These traditional methods are not effective in providing appropriate protection and can cause various issues and damage, for example, recently passwords have been easily and frequently hacked [162]. Multi-factor authentication relies on centralisation or trusting third-party services, which, as was discussed previously, have high security risks.

Recently blockchain has been used to improve protection against illegitimate access of several IoT applications without the need for centralised services. For example, Cha et al. [163] designed a blockchain gateway by integrating the blockchain in an IoT gateway to securely protect user preferences while connecting to IoT devices. This approach can raise the authentication level between the users and the connected devices. In addition, Sanda and Inaba [164] used blockchain technology with a Wi-Fi network to provide the authentication to the connected users and protect the network from malicious usage. The blockchain in this implementation was used to encrypt the communication and ensure security to the network. Therefore, the blockchain has the benefit of increasing the security of the authentication aspects.

3.4.2. Trust

Trust has been investigated in studies on the adoption of technologies that involve handling, storing, or processing sensitive information [165]. Blockchain technology ensures data persistence since it is stored in a distributed manner, which means many copies of the same ledger are shared, updated and synced among nodes. When a new peer joins the blockchain system, it can access a whole copy of the ledger, but it is more difficult to lose or destroy the

ledger. Unlike cartelised systems that own the ledger, blockchain provides data persistence by backing up the system, meaning that users have no choice but to trust the system.

3.4.3. Privacy

Privacy is an essential aspect for most of the systems. The majority of the researchers have taken advantage of blockchain technology to increase the level of privacy in the IoT environment and protect the individual private data being revealed [166]. For example, Kianmajd et al. [167] presented a framework that integrates blockchain to preserve users' privacy while using community resources. The framework highlighted that the decentralised environment of the blockchain can be used to increase the users' data privacy. In addition, Zyskind et al. [168] structured a personal data management platform in order to provide privacy for users. The study proposed a protocol that integrated with a blockchain to produce 'an automated trustless access-control manager'. The constructed platform achieved the privacy using encrypted data in the ledger and storing pointers to it instead of the transaction of the data itself to the network. Thus, personal data should be secured and controlled by the user and not be trusted to a third party.

3.4.4. Confidentiality, Integrity, and Availability (CIA)

Data confidentiality is an aspect of protecting data from unauthorised access. Since blockchain uses cryptography mechanisms, it offers confidentiality and protects data, such as bank account [156] and personal data [169], from parties that do not have permission.

Data integrity is another security aspect that is concerned with assuring and preserving the consistency, reliability, and accuracy of the data [170]. In other words, the data stored in the database should be kept from changing throughout its lifecycle. In this case, through the use of various cryptography mechanisms, blockchain technology provides data integrity and promises to protect data from unauthorised change [171], [172]. An accepted transaction in Blockchain network is stored by a distributed ledger in a block and then each block must reference its previous block to build a chain, any published block that did not refer to its preceding block would be rejected by other nodes. Banerjee et al. [173] combined the blockchain with IoT devices' firmware to maintain the integrity of shared data. Moreover, Liu et al. [174] implemented a framework for a data integrity service using blockchain to verify the integrity of IoT data without the need for a third party.

Data availability is one of many important terms in any system and means ensuring that the required data is available and accessible when needed [175]. One of the benefits of blockchain technology with a decentralised structure and distributed ledger is that it is resistant to outages.

To sum up, deploying blockchain technology to the proposed smart campus framework will provide more benefits by managing the problems associated with a centralised IoT architecture, especially from a security perspective.

3.5. Challenges of Adopting Blockchain in the Smart Campus Architectural Framework

The previous sections discussed the wide benefits of integrating blockchain technology into the novel smart campus architectural framework in relation to privacy and security aspects. However, blockchain technology is like any other advanced technology where challenges and issues can arise. Deploying blockchain to decentralise a smart campus framework has several major limitations that need highlighting. This section summarises these challenges and presents some current schemes for solving them.

Although security is considered the most important blockchain attribute, the threat of attacks from malicious parties cannot be eliminated. Since data is saved in a ledger which is distributed and public, there is a concern about data leakage. For example, in cases where 51% of a network's peers are malevolent, the blockchain's security could fail [66]. This concern may lead to hesitation in adopting blockchain technology in the smart campus setting and using it to store and share sensitive personal information, such as student credentials. However, in such an environment the peers in the blockchain network are known, which leads to the creation of a trustable system to share data safely. There are different models of blockchain classified depending on who can access the network and who can generate transactions, such as permission and permissionless networks. Each of these blockchain networks has its own features and characteristics. Therefore, it is necessary to choose a suitable model to match the environment system's needs. In this case, it is important to allow an authorised user to join the network in order to protect the security of the framework (for more details see chapter 4).

Despite privacy being enhanced by using blockchain technology, at the same time it is considered a concern. Many blockchain platforms use asymmetric key cryptography, which consists of public and private keys to ensure privacy and protect the ledgers. This feature comes

with the significant benefit of distributed systems in terms of protecting users' personal data and not allowing any unauthorised communication access. Therefore, the user has total responsibility to securely store his private key to avoid its loss. This is different from lost password authentication mechanisms where systems can offer a password reset function. A stolen or lost private key means the loss of all personal data associated with it. This issue may cause blockchain to be unacceptable for adaptation. In the case of developing educational institutes, losing the private key would affect the whole system [176]. Since this is a popular issue in blockchain technology, wide attention from practitioners and researchers has done much to protect and manage user authority keys. For example, Lei et al. [177] proposed a scheme for managing keys and securing communication in heterogeneous network such as IoT. This framework used the autonomous vehicle as a case scenario for testing. The scheme initialised keys and transmitted them using a rekeying algorithm to improve security. Panda et al. [178] developed a scheme for key management to improve secure communication among entities, as well as increasing efficiency. This approach assigned and authenticated the system's peers by using a one-way hash in the Ethereum platform. In addition, Pal et al [179] discussed different approaches to securing key management in the financial-based blockchain field where it can be studied, including local key storage, password-protected wallets, password-driven keys and offline key storage.

Blockchain's scalability is another concern that may be faced in distributed educational institute applications. This concerns the number of transactions and the amount of data that can be carried in a sole block. When the number of peers who join the network increases, this leads to an increase in the block size due to the increased size of the digital ledger [180]. Smart campus applications have to deal with and store a great deal of data, such as student records, staff information, data collected from smart devices, etc. This can cause the blockchain to grow excessively fast, which negatively affects the network's performance [144]. The growth number of the block size requires an increased number of transactions that need to be shared among peers to verify them and is thereby time-consuming and has low throughput. Bitcoin technology, as an example, can deal with only up to seven TPS [181], which is considered a major challenge if it is deployed in real-time systems such as smart campus where it needs to handle millions of transactions.

Many schemes and studies have focused on solving this problem. Some approaches used different techniques to enhance the validation phase in order to improve the slow speed of transactions. For instance, Bruce [182] developed a mini-blockchain method which can

increase the transaction validation process by reducing the number of transactions. This method removed the old records that were stored in the node to speed the validation process. Other researchers focused on compressing the block size. For example, Marsalek et al. [183] reduced the size of the blockchain by proposing a compressible blockchain structure. Instead of storing a whole blockchain copy in a client's device, the scheme reduced the size by encoded block data, thereby any later peers joining the blockchain have lighter data. A method was proposed to compress blocks in Bitcoin so that nodes could have less data size. This solution leads to a higher verification phase and lower processing memory and power. Wang et al. [184] increased permissioned blockchain scalability by reducing the block's header size using signature aggregation algorithms with parallel proof of vote consensus mechanisms.

Other approaches can increase blockchain scalability by using network sharding techniques where data is divided and stored by different peers. For instance, OmniLedger [185] was proposed to enhance the transaction speed of the blockchain-based UTXO data model by dividing the committees into different peers with each one storing and processing a different portion of data. Dang et al. [186] proposed AHL+, which developed sharding techniques to enhance the Hyperledger blockchain throughput. This scaling approach used a two-phase commit algorithm to support cross-shard transactions. ByShard [187] was developed using a multiple sharding transactions protocol to shard blockchain transactions that deployed the Byzantine consensus algorithm. Huang et al. [188] developed a resource allocation method for shared and scale permissioned blockchain networks in the PBFT environment.

Other schemes exploit off-chain to lighten the network and reduce the block size. Madill et al. [189] proposed a ScaleSFL which is a federated learning framework-based blockchain. This model uses an aggregation solution using off-chain to decrease PoC communication overhead. Aumayr et al. [190] designed Thora, a blockchain that supports the off-chain of multiple directional payments using payment channels. The off-chain channels allow users who do not have a creating payment channel to exchange cryptocurrencies; therefore, the transaction does not occur on the blockchain itself. In this thesis, the off-chain solution has been used to increase the scalability of the proposed framework to store data that do not need to be stored on-chain and to lighten the blockchain network (see chapter 5 for more details).

Since blockchain technology is a complex innovation, it may not be acceptable for deployment in smart campus areas. Blockchain is still in its early stages and is not well-developed in the field of education compared to areas such as cryptocurrencies and chain management [191],

which leads to challenges in adapting it into practice. Several intelligent campus applications-based blockchain technologies are proposed or have been developed; however, they are not widely applied in real life for reasons such as non-acceptance of the idea of decentralisation of their database, allowing public access, and not having control of their data. Lacking the comprehensive guiding framework to develop in such environments could be the main reason behind the concerns about deploying blockchain technology. Therefore, this study seeks to fill these gaps.

3.6. Summary

Recently, many researchers have focused on the study of developing smart and intelligent environments in many fields, such as smart cities, hospitals, and homes that mostly rely on IoT systems. The privacy and security aspects have been attracting research interest since they are considered the critical issues and challenges for connected IoT devices. This chapter analysed the various components of a smart campus, including architectures, platforms, and technologies with their limitations. Furthermore, a new smart campus architecture that combines the advantages of both the Internet of Things and blockchain technology was proposed. The framework considered as guide base to develop various campus services such as smart learning, smart building, smart parking etc. Moreover, this study discussed the security requirements for the proposed framework of a smart campus. Next chapter will be determined which blockchain platforms, based consensus algorithms, are most suitable for adopting into the framework particularly in a smart education environment.

Chapter 4 Analysis of Blockchain Platforms for Smart Education Environments

In the literature review in section 2.4, the thesis discussed the benefits of applying blockchain technology to various systems in a smart campus, particularly in education and learning fields, such as helping students distribute their credentials and certificates to prospective employers, disseminating published papers, building social collaborations and global interactions, and highlighting the benefits of using blockchain technology in such an environment. However, there is a lack of studies into which blockchain platforms, based consensus algorithms, are most suitable for adopting into a smart education environment.

Therefore, this chapter will address the third research question: ‘What methods can be used to develop a comprehensive framework for a smart campus implementation?’ It will act as a guide in selecting and developing a suitable blockchain platform for experimenting with applications, such as students’ transcripts, certificates, credentials or any other accomplishment data forms. To determine which blockchain platform to use requires discussing the quality requirements for blockchain-based smart education environments. The study then will review current and well-known consensus algorithms and provide a comparative analysis of all the platforms to choose a suitable one according to the smart education software requirements.

4.1. Quality Requirements for Blockchain-Based Smart Education Environments

Before analysing and comparing the current and common blockchain network frameworks, including their types and consensus algorithms applicable to the smart education environments, it is necessary to understand the software quality requirements for smart education systems. There is no doubt that different education use cases and scenarios lead to varying software

quality requirements. However, most use cases have common technical issues and requirements; focusing on and discussing these in this section to find a suitable blockchain platform for a smart education environment are needed.

According to ISO/IEC/IEEE International Standard [192] *quality* can be defined as “The degree to which a system, component, or process meets specified requirements”. In addition, a *quality requirement* is “a requirement that a software attribute be present in software to satisfy a contract, standard, specification, or other formally imposed document” [192]. The education system consists of staff, faculties, and organisations that coordinate to achieve educational goals [193]. According to Ila and Kitapci [194] to achieve high efficiency in such a system, security, privacy, performance, integrity, scalability, interoperability, and usability have been identified as necessary quality software requirements for a smart education system. Therefore, all requirements should to be fulfilled in this analysis.

First, **security** is one of the main requirements in this smart education framework. Hussien et al. [195] stated that data transformation amongst parties must be executed in a secured and trusted environment. Since academic data is sensitive, the system must support a reliable and secure technique to verify, store and share student’s data. The system should be identifiable every user and their actions. In addition, the data in the system must be shared only with the intended recipient, including faculty members, students, and authorities, while limiting the data distribution. A fitting blockchain platform must apply authentication features, restrict data access, and allow only authorised organisations to access the stored data under specific rules.

Second, closely associated to security is **privacy**. Logs in traditional blockchain mostly are publicly shared and disclosed by all peers in the network [196]. Privacy of academic data is a vital consideration in this framework. Academic data transactions should not be entirely transparent to every node, as with the traditional blockchain paradigm. Thus, a chosen blockchain platform should implement a robust set of privacy features.

Furthermore, **performance** or **operational cost** is another essential aspect to consider and includes the cost of associated transaction, storage, maintenance, and management of the academic data. Generally, integrating blockchain technology into most applications reduces the operational cost of traditional third-party storage, such as cloud computing [197]. However, this quality requirement is critical while comparing blockchain platforms. According to Holbl et al. [113], processing and verifying students’ academic data needs additional operational cost that should take it into consideration. Since each block in the smart education framework

contains qualitative and quantitative information, such as student records, learning outcome, course names, and course weights, a suitable blockchain platform is required to reduce computational costs and latency while storing and sharing data.

In addition, **data integrity** is another important aspect and a key requirement that need to be aware of for any platform dealing with data [198]. refers to the consistency and accuracy of stored data over its life-cycle [199]. Commonly, blockchain technology with hash functions, copied ledger, and mining algorithms ensure integrity and security of stored data against tampering and manipulation unlike a centralised structured system that more likely to attack from malicious parties and system failure or interruption- as discussed in section 3.6. However, different blockchain platforms that use different consensus mechanism may affect the system in different ways. Therefore, each transaction in this system should be accountable and verifiable, and integrity should be the desirable goal.

scalability is another requirement in the system. Scalability refers to the ability of a system to expand its existing capability of host volumes of data [199]. Generally, internet-scale systems should deal with a growing number of transactions and client requests. According to Mackenzie [200], network capability such as scalability is an essential requirement ambient intelligent systems along with power consumption and costs. Therefore, this smart education environment should support the rising number of students and their academic data as well as the different formats of this data, such as the student's scales, pictures, or pdfs, while remaining stable and usable.

Interoperability is an additional attribute that should be considered to ensure that the system can connect with other entities, such as processes, systems, software, or business units [201]. In the term of blockchain, Abebe et al. [202] defined interoperability as the transfer of values or data between ledgers with assurances of validity. Developing a framework that integrates with blockchain technology designed to be decentralised, with each entity able to communicate, does not mean that the system will not have interoperability difficulty. In addition, Khan et al. [203] stated that the framework should support smart contracts to allow invokes and calls among entities to accomplish blockchain interoperability. Therefore, comparing different blockchain platforms that deploy a smart contract is an important aspect to achieve interoperability.

Lastly, **usability** is another important requirement which focuses on develop a software that is easy to operate and control [204]. In the case of handling student's data, several developed

schemes require individuals in the blockchain network to manage their data, including public and private keys, to generate cryptographic signatures and authorise access to their private data. Nevertheless, a user-friendly interface should hide the back-end complexity of managing user's key pairs when develop applications. Additionally, completeness is a part of the usability requirement that should be considered. This system, known as the end-to-end system, provides automation. The framework should allow faculty members, students, and authority institutions to easily upload, request, transfer, and validate data. For more automation, the suitable blockchain platform utilisable for the smart learning framework should support the execution of smart contracts. Not all blockchain platforms support smart contracts. However, it can facilitate the automation and programming of the rulesets that manage communication among stockholders. Since blockchain technology is still in its early stages, it is crucial to choose a fitting platform to be aware of the availability of programming languages and tools for developing smart contracts.

Smart education environment software quality requirements can be achieved by utilising blockchain technology with different properties (as discussed in section 3.6), such as decentralisation, security, immutability, and autonomy. Therefore, if a suitable blockchain platform is selected correctly in a smart education system, it can help enhance the system's security, ensure the privacy and integrity of data, and encourage individuals and organisations to share data. The next sections will study blockchain platform types and their characteristics, which are crucial to deciding which blockchain to adopt and comparing their consensus algorithms based on the requirements of smart education systems.

4.2. Permissioned vs Permissionless Blockchain Networks for a Smart Learning Environment

Blockchain technology networks can be classified into two different models: permissionless and permission networks [205]. A permissionless network is a network where any node can publish a block while a permissioned network is a network where a particular node can publish a block. In other words, a permission blockchain network controls the network, and a permissionless blockchain network means any peer can participate from the public internet – see figure 4-1.

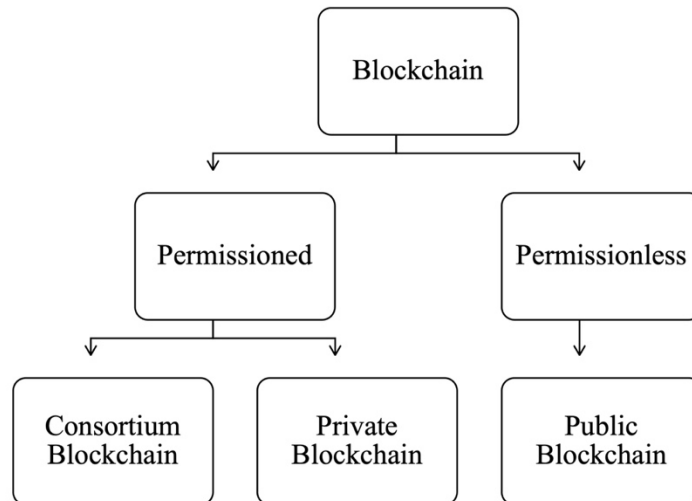


Figure 4-1 Blockchain network classifications

According to the figure, a public blockchain is a type of permissionless blockchain network, while private and consortium blockchains are types of permissioned blockchain networks. Each of these blockchain networks has its features and characteristics (see figure 4-2) that will be discussed in more detail in the following sections.

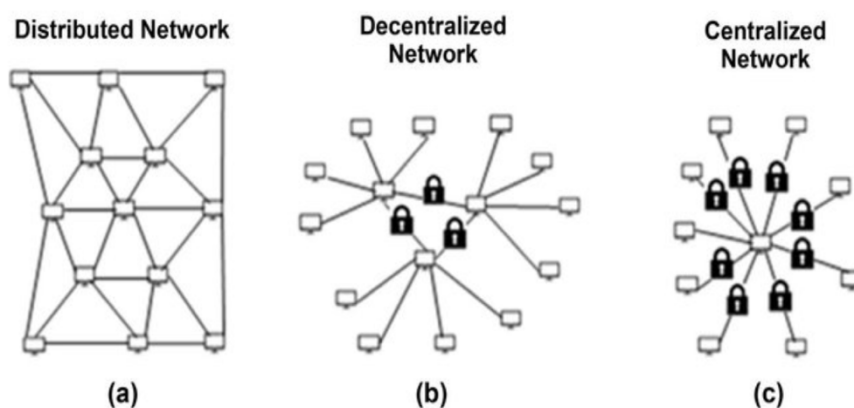


Figure 4-2 (a) Public blockchain network; (b) Consortium blockchain network; (c) Private blockchain network

4.2.1. Permissionless blockchain network

A permissionless or public blockchain network sees a decentralised ledger opened to any peer to join the network and publish a block without authority or permission, as can be seen in figure 4-2 (a). This network is mostly open-source software and available to any user to download without permission and may execute millions of devices. Additionally, the network allows any peer to join, publish blocks, issue and validate transactions, and read the blockchain. Thus, any node in the permissionless blockchain network can access the ledger to read or write without a required user's identity.

This type of blockchain network applies consensus or agreement algorithms (which will be discussed later in more detail) that run in the blockchain's protocol to prevent malicious peers from accessing the ledgers. Proof of work is one of the well-known consensus algorithms, and proof of stake is another approach. Since every node in the network applies a consensus mechanism to verify the transactions to increase the security, this leads to consuming a vast amount of resources, such as electricity and storage costs, in addition to producing low transaction throughput. Examples of permissionless or public blockchain networks are Bitcoin [89] and Ethereum [75] ; they are widely used in finance and cryptocurrency areas.

4.2.2. Permissioned blockchain network

A permissioned blockchain network means only an authorised user can publish blocks. The network has restrictions on who can access the blockchain and issue transactions. Different restriction rules can be applied to permissioned blockchain networks; it may allow any user to read the ledgers, restrict it to only authorised nodes or permission users, or it may allow any user to issue transactions in the network. A permissioned blockchain network can be open or closed source software.

As permissionless blockchain network, permissioned blockchain network applies consensus algorithms such as Byzantine-fault tolerant to publishing blocks. However, they do not require maintenance or expense of resources. The particular reason for this is that a permissioned blockchain network requires a user's identity to participate in the network. Therefore, the network has a level of trust with its participants since they are authorised. Additionally, these features of permissioned blockchain networks make consensus algorithms faster and result in lower operational costs (as discussed in more detail in the next section).

A permissioned blockchain network is suitable for organisations that are required to control their blockchain or organisations that are working together but do not fully trust each other. These organisations can establish the consensus algorithms based on which trust level they require. Besides trust, transparency can be achieved in this network and might be an asset to improve business decisions.

Private and consortium blockchain networks are types of permissioned blockchain networks. The former is a network where only invited and selected nodes have permission to access the network. It has a distributed feature, although it is partially decentralised since a peer has to control permissions to validate transactions, change the rules, and write and read the ledgers,

as can be seen in figure 4-2 (c). A reduced number of involving nodes helps efficiently consume resources and transaction throughput more than a public blockchain. Nevertheless, private blockchain networks affect security and trust because the trusted peer can simply be compromised and reduce the transparency and immutability of the network, leading to it being highly centralised (still, it benefits more than traditional databases). Ripple [206] and Eris [207] are examples of such a network.

On the other hand, consortium blockchain networks combine the advantages of private and public blockchain networks, as can be seen in figure 4-2 (b). In a consortium blockchain, a selected number of peers have permission to join the network, however, all the invited nodes have rights to read and write to the ledger and verify transactions, which is similar to a public blockchain's decentralised nature. In terms of transparency and immutability, its structure falls midway between the two previous chain models. Its features lead to enhanced efficiency compared to the public blockchain network. Additionally, nodes in the consortium blockchain compute their role by either endorsing peers responsible for executing the smart contract or committing peers responsible for committing the transactions, which helps to improve the network performance and throughput, unlike public blockchain networks that suffer from redundant computations. Furthermore, this network, compared to private networks, is more trustworthy and less vulnerable to threats of security since the network is not controlled by a single node to validate or authorise, write, and read to the ledger. Hyperledger [105] and Corda [208] are examples of consortium blockchain networks. Table 4-1 summarises the above-discussed blockchain network types and their features.

Table 4-1 Summary of blockchain network types with their features

Criteria	Public (permissionless)	Private (permissioned)	Consortium (permissioned)
Network type	Distributed	Centralised	Decentralised
Peers joining the network	Any peer	Authenticate peers	Authenticate peers
Require membership services	No	Yes	Yes
Validate transaction by	All peers	Single peer	Selected peers
Trust level	Not trusted	Trusted/compromised	Maximum trust
Throughput rate	Low	High	High
Efficiency	Low efficiency	High efficiency	High efficiency
Energy consumption	High	Low	Low

To answer the addressed questions presented earlier in this chapter regarding which blockchain network platform can be used in the smart learning framework, software quality requirements have been discussed, and blockchain types have been presented with their features. Additionally, Pahl et al. [209] developed a flow chart framework to assist in the decision-

making process to adopt a suitable blockchain platform, as shown in figure 4-3. The framework has been used to help find an appropriate blockchain network platform. The result is presented below:

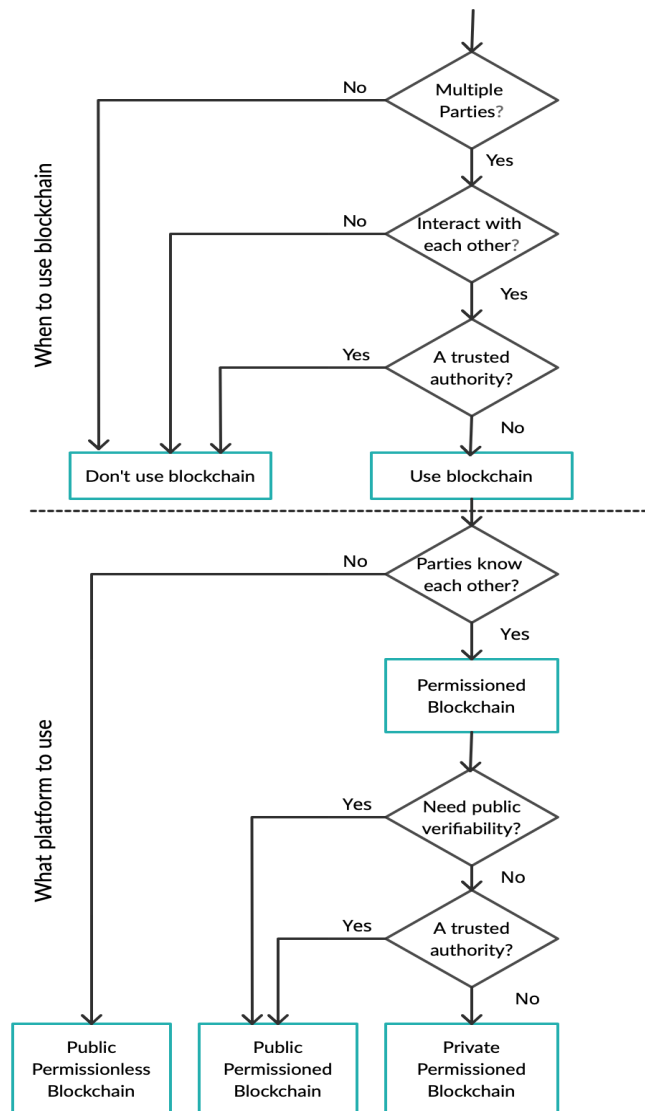


Figure 4-3 Decision flow to adopt a suitable blockchain platform [209]

The framework helps answer two vital questions before using blockchain technology for any software: 1) whether blockchain technology is needed for the software; 2) if yes, which blockchain platform is more likely to be used? The flowchart is split into two parts; the first part aims to answer the former question, while the second part aims to answer the second question by examining which blockchain properties are needed.

The first upper half of the flowchart addressed questions to help developers decide whether blockchain technology is needed in the application. The upper half checks if the system

requires multiple parties. The involved parties either have the same roles, such as validators, writers, and readers, or different functions, such as writers and readers. The authors' point of view is that if a party needs to fulfill all the roles then blockchain technology is not required to manage data. In this case, numerous parties are involved in the system, and each has a different role. For example, university faculties can validate transactions and read and write to the ledgers of a student's data, a student can validate transactions and only read the ledgers, and a third authority, such as employers and insurers, can only read the ledger. Thus, the answer to this question is yes; the system has multiple parties with different roles. Then, the second criterion is if the parties interact with each other. If not, then a simple log is needed to record data by an independent party. In the study, all parties interact with each other. The last criterion of this part is a trusted third party. Blockchain technology is designed to eliminate third parties, such as cloud computing, since it leads to numerous issues, such as single point of failure, DDoS, and DoS, as well as increased security, privacy, and integrity of data. Therefore, blockchain technology is needed for integration into the framework.

After answering the first question, the bottom half of the flowchart leads to answering the second question, which determines the type of blockchain network that can be used. The first criterion is whether parties know each other (e.g., anonymity of parties). If a system interacts with anonymous users, then a permissionless public blockchain platform is the best fit for such an environment. However, the system interacts with users that know each other. Thus, a permissioned blockchain network is recommended as it has access restrictions to the network of this group and provides a faster consensus algorithm process and a higher transaction throughput rate. The next questions select between the two types of permissioned blockchain networks, namely private or consortium. The paper [209] named them 'the public permissioned' blockchain networks. The next criteria are related to public verifiability and read access. In the framework, public reading is needed in some cases, such as when a student wants to share his/her certificate with any party out of the network. Therefore, after examining the flowchart, a consortium blockchain network is the best fit for the smart learning framework. The next section compares consensus protocols.

4.3. Comparing Consensus Mechanisms

Since a blockchain network is a distributed ledger and not controlled by an authorised single node, verifying a transaction needs to reach a consensus or agreement among the majority of

network peers. Thus, consensus algorithms are a mechanism utilised by various platforms of blockchain networks to ensure the consistency and integrity of stored data across geographically distributed peers. There are several different consensus algorithms between novels developed by researchers and existing open-sources implemented in platforms. This section focuses on existing adopted consensus algorithms that will be reviewed and compared to find a fitting blockchain platform for the system. The reasons for this stage are that many blockchain platforms have pluggable features and can run numerous different consensus mechanisms. Additionally, there is no perfect algorithm, and each one has its pros and cons. Therefore, studying consensus protocols and finding a suitable one for the system is crucial since they work behind blockchain platforms.

To find a fitting blockchain platform, it is necessary to discuss different consensus algorithms with their functionalities and characteristics. To determine which consensus algorithm should be used on the system and to meet the requirements in section 4.2 requires higher fault tolerance and lower verification delays. There are hundreds of consensus algorithms categorisable into three groups: compute-intensive consensus algorithms, capabilities consensus algorithms, and voting consensus algorithms:

4.3.1. Compute- Intensive Consensus Algorithms

This type of consensus algorithm allows a contributing node to generate, add, and verify a new block in the blockchain. Additionally, these algorithms are energy-hungry, meaning miners in the network use their computing energy to win the lead to propose and mine a block in the next round. This type is designed to be applied in public blockchain networks, particularly cryptocurrencies such as Bitcoin [98], Dogecoin [210], and Litecoin [211]. Examples of compute-intensive consensus algorithms are proof of work (PoW) [98], Prime number PoW [212] and Delayed PoW [213]. Since a consortium blockchain is a suitable blockchain network for the system (as discussed in the previous section) and these consensus algorithms alongside it are developed for a public blockchain network, it requires high network bandwidth and high latency. Therefore, compute-intensive consensus algorithms are not a rational choice for the system.

4.3.2. Capability Consensus Algorithms

When the previous type of consensus algorithm consumes high energy, several consensus algorithms were developed to enhance these cons and elect a miner in the blockchain network

based on non-computing capability. Numerous capability factors can be calculated to determine a miner of the network, such as the amount of digital currency owned by the miner, the amount of storage the miner has, and the miner's trust in the network. For example, Proof of Stake (PoS) [214], Delegated PoS [215], Proof of Elapsed Time (PoET), Proof of Stake Velocity [216], and Proof of Burn [217]. Since capability consensus algorithms were developed to address the issue of high-power consumption in compute-intensive consensus algorithms, they have the same use for public blockchain networks, such as cryptocurrency, that are unsuitable for the framework. However, there are some capability consensus algorithms that can be applied for a permissioned blockchain, such as PoET, which will be discussed in the next section.

4.3.2.1. Proof of Elapsed Time (PoET)

Proof of Elapsed Time (PoET) is a consensus algorithm to improve security developed by Intel and relies on a new processor instruction set named Intel Software Guard Extension (SGX). The PoET consensus algorithm has the same concept as the Nakamoto consensus mechanism processing in Bitcoin. Both algorithms need to elect a leader to propose a block in each round to be added to the chain. However, the difference lies in the method of selecting a leader. The Nakamoto algorithm uses a lottery mechanism to choose a leader, which consumes power to achieve proof, while PoET elects a leader by using the Intel SGX capability as a Trust Execution Environment (TEE).

PoET is deployed on the Hyperledger Sawtooth platform, which is a software framework founded by Intel to create distributed ledger networks for a range of use cases [218]. Every validator node in the Sawtooth during each round requests a wait time from a trusted function in the SGX. The shortest waiting time is assigned for a validator selected as a leader for that round. In other words, electing the current confirming peer in the consensus utilises random waiting times for confirmation peers. Then the leader, e.g., the winning validator, can create a block that contains a series of transactions. Additionally, other validators validate the block validity before it is added to the blockchain.

Therefore, the PoET consensus algorithm has helped the Sawtooth network to reach massive scalability with less computational power since it does not need to solve intensive cryptographic puzzles. Additionally, its high throughput and low latency may be applied to the system. However, the main disadvantage for this algorithm is the required specific SGX hardware as well as its dependency on Intel, which developed SGX hardware and goes against the

blockchain concept philosophy of being decentralised and not relying on third parties. Thus, PoET is not the best fit for the framework.

4.3.3. Voting Consensus Algorithms

This type of consensus algorithm is deployed in a private and consortium blockchain network – the latest one has been chosen for the network and focused on. There is no need for participating nodes to generate and add a new block in the chain, and only authorised peers can participate in the process of consensus algorithm to generate and validate a new block. The voting consensus algorithm elects a miner to generate a block by using a voting system that ensures fairness among delegates based on an existing agreement. This consensus does not consume as much energy as compute-intensive consensus algorithms due to using a competitive approach to select the miner. It also eliminates the problem of the rich becoming richer, as seen in capability consensus algorithms, since the wealth dominance is selected to be the miner. Thus, it is well-suited for areas beyond cryptocurrencies. These algorithms are developed to address Byzantine fault tolerance. This happens when some of the nodes in the network behave maliciously or independent nodes fail. Byzantine fault tolerance in distributed systems reaches the desired consensus on the network despite nodes acting maliciously or failing. Voting consensus algorithms can be broadly categorised into two groups: Byzantine fault tolerance (BFT); and crash fault tolerance (CFT) [90]. The former mechanism results from the problem of the Byzantine general discussed in more detail in reference [219]. The idea of BFT in a blockchain network is to avoid complete failure of the network by reaching agreement among nodes on a single state, assuming that some of the peers could be failures. There are many protocols-based BFT consensus; the well-known one is practical Byzantine fault tolerance, deployed for numerous blockchain platforms, while the latter one was developed to prevent the case of crashing/failing nodes due to software or hardware failures. Raft and Kafka are the famous algorithms-based CFT consensus.

After viewing different types of consensus algorithms applied behind blockchain platforms, voting consensus algorithms are the best fit for the system since a permissioned blockchain is considered more trustworthy and confined. Therefore, their blockchains do not require mining or hashing procedures. Instead, they need to run message-based consensus algorithms, which are lighter and fitting for areas beyond cryptocurrencies. To determine which consensus mechanism can be used, well-known voting consensus algorithm types under both BFT and CFT schemes will be compared in the following sections.

4.3.3.1. Practical Byzantine Fault Tolerance (PBFT)

In 1999, Castro et al. proposed Practical Byzantine Fault Tolerance [220]. The consensus algorithm is based on a Byzantine generals problem. The idea of the protocol is one central authority node rolling as a leader and selecting a group of nodes to be the backup nodes. In the network, all nodes should communicate with each other to reach a consensus or agreement, and all authority nodes should have the same copy of the ledger. The number of malicious nodes in the total nodes in the network deployed in the PBFT mechanism must not be greater than or equal to $n/3$.

Generating a block in PBFT consensus, known as a view, requires four phases in each round. First, the leader node receives a request sent from the client to perform a transaction and generate a new block. Second, when the leader node receives all the request transactions, he will group them in a block to be broadcast to other nodes in the network. Then, the nodes verify and validate the transactions in the block. Additionally, each node computes the block's hash to be sent to other backup nodes. Finally, each node waits to receive the same hash from at least two-thirds of nodes or $f+1$, where f represents the number of faulty or Byzantine nodes [221]. When the node collects the same hash, the new block will be added to the blockchain ledger of the node.

Several blockchain platforms deploy PBFT in their consensus, such as Hyperledger Fabric v 0.6, Hyper ledger Iroha, and BigchainDB. The throughput transaction of PBFT consensus is better compared to compute-intensive consensus algorithms and capability consensus algorithms. However, PBFT requires an authority to elect the backup nodes and a leader, which leads to it being less decentralised. Therefore, PBFT is suitable for permissioned blockchain networks since they select their authority nodes. However, when the number of participating backup nodes increases, scaling becomes an issue because of the high communication messages required [222]. The increased number of transferred messages in PBFT consensus could lead to a rise in computing energy because of communication and network overhead. Additionally, PBFT can encounter Sybil attacks communication and network overhead. In addition, PBFT can be attacked by sybil attacks [223] where one adversary can create numerous faulty nodes without any certificate authority or control a fraction of the network to taint the outcome of the consensus to their favor. Therefore, PBFT has low latency, low computational overhead, and high throughput, which is suitable for the system. However, the high communication overhead in the network leaves it unsuitable for large networks and could be applied to small educational systems.

4.3.3.2. Delegated Byzantine Fault Tolerance (DBFT)

The DBFT mechanism was proposed in 2014 by the NEO blockchain platform [224]. The algorithm as PBFT, both utilise the voting process to select the nodes. However, DBFT does not require all nodes to participate in adding a new block, which adds more scalability to the network. A leader node in the DBFT refers to the speaker and the backup nodes are the delegates. A voting system is used to elect the delegates from whom a speaker is randomly selected. The next step sees a block generating and validation process similar to PBFT. Since the delegates are elected based on voting, there is a possibility that each participating node can vote for itself to be selected. Therefore, all the participants will be elected as delegates, which will lead the network to suffer from communication overhead issues.

To reach the consensus in DBFT, the total number of failing or malicious nodes should be less than $(2n-1)/3$ in the network, where n is the total node. The consensus has many features that could be used for the system. However, besides the issue of communication overhead, the issue of Sybil attacks is the same here as PBFT. Additionally, the average latency in DBFT to generate a new block is 15 seconds, which is more than PBFT. Therefore, the DBFT consensus algorithm is not suitable for the system.

4.3.3.3. Stellar Consensus mechanism

The consensus was proposed by Mazieres and uses the Federated Byzantine Fault Tolerance (FBFT) consensus, which is a variant of PBFT, as the backbone of the algorithm [225]. Nodes in FBFT are intersected into groups named federates, and each group executes a local consensus algorithm. This approach allows everyone to join the network and participate in the agreement mechanism, i.e., the consensus opens to the public. Stellar consensus uses the same concept, also referred to as Quorum, by grouping nodes into sets that reach an agreement. A Quorum slice is a subset of a Quorum that helps one particular node reach the consensus process. Common nodes can be shared and overlapped among Quorum networks in a process known as intersection.

Stellar consensus needs two phases for the voting process, namely nomination and ballot protocols. When executing the former step, candidate values, which are new values, are proposed for consensus and broadcast to all participating nodes in the Quorum network. Each node will vote for a value from the candidate values. Quorum slices then influence each other with the help of Quorum intersection. Then, the later stage is initiated to vote for either aborting or accepting the obtained values from the previous stage using federated voting. At the end of

the process, aborted ballots for the current slot are rejected. The Stellar mechanism reaches the agreement stage when each group agrees then broadcasts to the other groups in the network.

The method has low computational and high throughput requirements, which are desirable for the system. However, according to Pahlajani et al. [226], Stellar has security issues if the network selected is not a proper Quorum slice. Such a situation occurs because each subset group runs a local consensus among its nodes before the agreement propagates to the rest of the network using intersections. Additionally, the latency of this approach is not low compared to the other consensus. Therefore, this method is not suitable for the system.

4.3.3.4. Raft consensus mechanism

The communication overhead issue presented in PBFT is a common issue in BFT-based consensus. It is eliminated in the CFT-based consensus by allowing communication between the leader and the backup nodes without the need for the backup nodes to communicate with each other [221]. A well-known CFT-based consensus is Raft, proposed by Ongaro et al. in 2014 to provide security on a network against node crashes without addressing the safety situation against malicious attacks. The consensus provides correct operation if more than half the nodes in the network perform typically at a given time, e.g., $t < n/2$ where n is the total participating nodes and t is the number of crashing nodes. Each node in the network at any given time can be in any of the following states: follower, leader, or candidate. In the follower state, the following node simply and passively replies to the requests from the candidates and the leader. In the leader state, the leader is the node that receives transactions from the clients and generates the log transactions. There is a single leader in the Raft consensus. In the candidate state, a new leader is elected by the candidate node. The requests in the network have been carried out by using RPC calls: Request vote calls are used to select a leader and the candidate who initiates them during an election; Append entries calls are used for the replication of logs and the leader responsible for initiating it; and Install snapshot RPC calls are used to send a replication log from leader to followers.

The Raft consensus mechanism solves the issue related to crash tolerance. However, it does not ensure network safety against malicious tolerance. The consensus can reach the agreement of up to 50% of the crash fault. Since all nodes in the system are known and have permission to access the network, malicious tolerance is low, and the crash fault is most crucial in the framework. Additionally, this consensus has low latency and high throughput, which means it

is suitable for the system. Several blockchain platforms deploy Raft in their consensus, such as Hyperledger Fabric v 1.0 and Quorum.

Table 4.4-1 presents a comparison of previous consensus approaches and their suitability. (Good) is the least suitable, and (Excellent) is the most suitable for this framework. Consequently, Raft has high scalability, unlike PBFT consensus algorithms that need two nodes to communicate multiple times to confirm a transaction. When the number of nodes increases, more communication is required, which decreases the synchronisation speed. Additionally, Raft has low latency compared to DPBFT and Stellar and does not require specific hardware, such as PoET. Thus, the Raft consensus algorithm is the most apt mechanism for the framework.

Table 4-2 Summary of the comparison of consensus algorithms for an education environment

Criteria	Consensus Algorithms				
	PoET	PBET	DPBFT	Stellar	Raft
Latency	Low	Low	Medium	Medium	Low
Computing overhead	Low	Low	Low	Low	Low
Scalability	High	Low	High	High	High
Throughput	High	High	High	High	High
Adversary tolerance	N/A	<33% Fault replicas	<33% Fault replicas	Variable	<50% Crash fault
H/W required	Yes	No	No	No	No
Our suitability	Good	Very Good	Good	Good	Excellent

In the next section, the deployed Raft consensus mechanism into blockchain platforms will be discussed in more detail to select a fitting platform for the system.

4.4. Blockchain Platforms

An analysis of existing consensus algorithms considered the backbone of blockchain implementation has been studied in the previous section to choose the most suitable consensus mechanism for the smart learning framework. The analysis found that the Raft consensus algorithm is the most appropriate one to use in this case because most blockchain platforms have pluggable features. Therefore, different existing consortium blockchain platforms deploying a Raft consensus algorithm will be addressed and analysed according to their

characterised design and transaction flow. Quorum, Coda, and Hyperledger Fabric are well-known permissioned platforms that deploy the Raft consensus algorithm.

The Quorum blockchain platform [227] is a permissioned network that forked from Ethereum. The platform was founded by JP Morgan after changing core aspects in the Ethereum platform, such as privacy of transactions, the consensus algorithm, permissioned participants, and eliminating transaction fees. The Corda platform [228] is a permissioned blockchain developed by R3, and it is a semi-open-source blockchain. The platform uses a notary pool to reach consensus, and it is widely used in financial sectors since it is similar to traditional banking systems. Additionally, Hyperledger Fabric [229] is a permissioned network founded by Linux Foundation. It is an open-source blockchain proposed to meet the needs of business consortium networks. Additionally, the Fabric and Quorum platforms are open-source, whereas the Corda platform is semi-open-source, which means some part of the source is closed to provide secrecy in relation to sensitive business applications. Theoretically, the platform founders (JP Morgan, R3, Linux Foundation) all claim their frameworks have high performance, security, scalability, usability to develop a consortium chain, and the capacity to develop smart contracts. Therefore, they will be discussed in more detail in the following sections to compare, differentiate, and analyse them with additional criteria, such as architecture, focus, language, transaction rate, and so forth.

4.4.1. Quorum Platform

The Quorum platform [227] was founded by J.P Morgan and it was created based on the Ethereum blockchain. Ethereum is a permissionless blockchain network, which is a decentralised ledger that is open for the public to join and publish a block without needing permission. Therefore, each node participates to verify the transaction by running the PoW consensus algorithm. To enhance Ethereum efficiency, Quorum has been inherent as a permissioned version that has a limited number of nodes that run smart contracts and verify transactions. It is an open source that is mainly used for banking sectors [230]. In addition, it has a pluggable feature that allows it to execute different consensus protocol implementations. Quorum architecture consists of three main components [231]: Quorum node, which is responsible for communicating with users using a command-line interface as well as adding new blocks to the blockchain; transaction manager, which plays a role in communication with other transaction managers to verify transactions, and enclave, which is responsible for cryptography, including generating symmetric keys, decrypting and encrypting data. When a

node initiates a transaction, all members involved vote on it, and it is then broadcast to all participants in the network if it has the majority of votes. Once transaction managers in the network receive the transaction, they will start matching the hash to validate the transaction. If it is valid, the transaction will be stored in the chain, otherwise it will be rejected.

4.4.2. Corda Platform

Corda [208] is a distributed permissioned globe blockchain platform that was introduced by R3. The platform is intended for processing and recording financial agreements, and most of its scenarios are drawn for the financial services [232]. Therefore, the measurable data, such as stock price and currency, is suitable for this platform. With the support of smart contracts in Corda, trading activities and financial workflow become more organised and automated. It enables interoperation among various organisations and systems on a single network [230]. Since it is a permissioned blockchain, each peer needs to validate their identity to access the network. The peer who is responsible for distributing the certificates and validating identities is named Doorman. Corda also has notary nodes; they play a role in validating the transactions or uniqueness without broadcasting globally, i.e. the transaction information is not stored to all peers' ledgers in the network, only to the interest nodes.

To add a transaction to the Corda blockchain, reaching a consensus is required. There are two parts of a consensus that are needed to update the ledger: validity and uniqueness. Validity is when each peer validates the transaction before signing it by checking all the necessary signatures and assuring that the associated smart contract is verified successfully. Uniqueness is related to notaries, which check the input states of the transaction to ensure they have a unique consumer, i.e. for the duration of the state's lifecycle, notaries ensure that the state is not used as an input for more than one transaction to avoid double-spending.

4.4.3. Hyperledger Fabric Platform

The Hyperledger ecosystem, founded by the Linux Foundation, created the first consortium blockchain named Hyperledger Fabric [229]. It is an open source blockchain project that has been designed to fit different use cases in businesses and governmental organisations by ensuring privacy of a blockchain network. Maintaining multiple distributed ledgers within a system is a main feature of the platform. In addition, a crucial strength of the platform is its pluggable features and modular design. For instance, fabric ledger data does not depend on a specific format lead to be suitable for various case scenarios. The fully pluggable feature leads

the consensus mechanism to use different algorithms. Therefore, different situations use different consensus mechanisms.

Hyperledger Fabric consists of six main components: peers, orderer, endorser, shared ledger, chaincode and member service provider gossip network protocol [229]. Each of these components is involved in the consensus algorithm to validate transactions. The special entity of the Fabric consensus process is orderer, since it is responsible for generating a new block and attaching it to the ledger in the appropriate order. In addition, endorser is another entity that plays a role in the network. It is responsible for endorsing and validating a transaction, as well as checking whether any part of the network may execute a certain action in a ledger. Other participating peers involve themselves in the network by creating transactions. Since Fabric is a permissioned blockchain network, all its entities are authenticated and registered via a member service provider which provides roles by authorising and managing all participants' identities in the network. Thus, using an identity provider leads to an increase in security policies to allocate actions to entities within a specific ledger.

To sum up, table 4-3 presents a summary of the comparison of blockchain platform for education environment as the following:

- **Focus:** Each blockchain framework is designed with a primary focus. Quorum and Corda are designed for the finance sector, while Hyperledger Fabric is built for business-to-business (B2B) and general purposes.
- **Languages:** Supporting more general and popular languages will lead to more developers, and the system will become more adaptable. Quorum supports Solidity as it is forked from Ethereum. Corda supports Java and Kotlin, while Hyperledger Fabric support Go, Java and NodeJS. According to Cai et al. [233], stable and flexible implementation is important factor to be aware of while selecting a prospective blockchain platform. Depend on [234], JavaScript is the most popular language among professional developers.
- **Transactions rate:** Measured as transactions per second (TPS), Corda tolerates 170 TPS, Quorum handles a few hundred TPS, while Hyperledger Fabric can reach 2000 TPS.
- **Architecture:** Quorum is designed as order-execute architecture, meaning transactions are ordered first in the architecture and then executed sequentially on all peers in the

same order, affecting the system throughput [235]. A smart contract in this architecture uses non-deterministic transactions. All peers in the blockchain network hold the same state; they operate the same transaction and return the same outcome to support domain-specific languages, such as Solidity, to design smart contracts and do not support generic languages. On the other hand, Corda and Hyperledger Fabric use architecture based on the execute-order-validate (EOV) approach [235], which allows for transaction execution before ordering them to tolerate non-deterministic smart contracts in the blockchain network.

Table 4-3 Summary of the comparison of blockchain platform for education environment

Criteria	Blockchain Platforms		
	Quorum	Corda	Hyperledger Fabric
Founder	JP Morgan	R3	Linux Foundation
Focus	Banking Sectors	Financial industry	Enterprise & B2B
Languages	Solidity	Java, Kotlin	Go, Java, NodeJS
Transactions Rate	Few hundred TPS	170 TPS	2000 TPS
Architecture	Order-Execute	EOV	EOV

In conclusion, after comparing the three aforementioned platforms including Corda, Quorum and Hyperledger fabric, they all theoretically could be utilised for the framework since they all have high performance, security and scalability, in addition, they all can be used to develop a consortium chain, they all deploy Raft consensus algorithm, and they all have the possibility to develop smart contracts. However, due to Hyperledger Fabric overall positive reputation, active community, and considerable publicity make it the most suitable blockchain platform with a Raft consensus algorithm for this smart educational framework. In addition, a Hyperledger blockchain is widely used for businesses and enterprises. It is designed to support pluggable implementation of components delivering high degrees of confidentiality, resilience, scalability, and low latency. Hyperledger has a modular architecture and can be used very flexibly. Moreover, modular consensus protocols are been used, which permit a user to trust models and tailor the system for particular use cases. This platform runs smart contracts or chaincode, which is an executing programmable code that allows participants to write their own scripts without a middleman [236]. Next section will provide the discussion in detail of the results of using Hyperledger fabric to the system.

4.5. Results of a Suitable Blockchain Platform for Smart Education Environments

As a result, the Hyperledger Fabric blockchain platform, with the permission features of the network, provides various benefits to the smart education framework, particularly in sharing and storing data.

- **Transparent network:** This smart education framework allows private channels between users to enable certain information to travel transparently and freely. For instance, a channel will involve the issuer, the verifier and the owner in order to allow a transparent flow of data among them.
- **Uniquely Identifiable documents:** A document such as a student's grades, credentials, and certificates will be integrated with a unique identifiable hash of the document's participants, e.g., the owner, verifier, and issuer, that lead to being immune to tampering. If any changed happened to the hash, the Hyperledger on the verification stage will refuse the document since the hash not match with the saved original hash.
- **Customised access level:** As the Hyperledger Fabric platform is a role-based blockchain, it requires customisation of the access level to the documents in the framework.

4.6. Summary

This chapter delivered a comprehensive comparative analysis of suitable blockchain platform-based consensus algorithms. Since consensus algorithms are behind blockchain platforms, most of the performance and feature aspects of blockchain platforms are affected depending on the used consensus. Therefore, it is important to find an appropriate consensus for the architectural framework. To evaluate the different consensus algorithms that are most used, they should meet the quality requirements of the smart educational framework. The target consensus should have high security and scalability as well as low computing overhead and latency. The high throughput aspect is essential to enhance the operational costs. However, low latency is a more important since high throughput is the most significant feature for cryptocurrency, while low latency is an important feature for networks beyond cryptocurrency. As a result, the Raft consensus algorithm has been chosen. In addition, from the chapter

analysis of blockchain platforms that were designed as permissioned blockchain to support multiple business areas and deploy Raft consensus, the Hyperledger Fabric was the most suitable platform. The proposed framework will be designed in the following chapter.

Chapter 5 A Smart Campus Framework Architecture Design

After proposing a novel framework for a smart campus, which was explained in Chapter 3, and determining a suitable consensus algorithm and blockchain platform for education environment, as discussed in Chapter 4, evaluating the framework is the next step. In order to do so, it is necessary to design the architecture for the implementation stage, which helps to continue answering the third research question. In the previous chapter, the study found that Hyperledger Fabric is an appropriate platform for the framework, since it delivers decentralisation, data integrity, and security. Therefore, in this chapter, the architecture and design of the novel framework is explained. Then, based on this design, implementing and evaluating different use case scenarios can be done.

First, the chapter provides an overview of the technologies stack required for the framework and the reasons behind on-chain and off-chain data storage. Then, each on-chain and off-chain technology used will be discussed independently with more detail regarding its components, design, and transaction flow.

5.1. Technologies Stack

In this section, the high-level architecture of the design and the main technologies used was proposed. Figure 5-1 shows the technologies stack. Hyperledger Fabric is used as a blockchain platform and Inter Planetary File System (IPFS) as decentralised off-chain storage. The model is immutable and content addressed, as well as ensuring security, privacy, and integrity of the data.

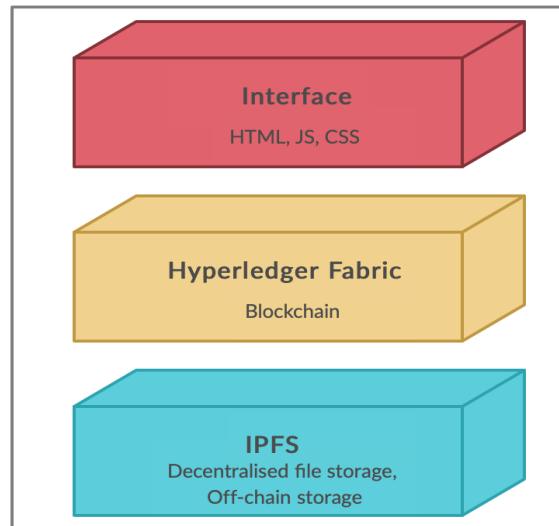


Figure 5-1 Technologies stack

On-chain: As was addressed in the previous chapter, Hyperledger Fabric is an ideal blockchain to store and share sensitive data while maintaining privacy and confidentiality within the network, since it allows only authorised participants to access the network [237], [238]. Because participants are identifiable and limited, Hyperledger uses lightweight consensus protocols rather than permissionless blockchain, which leads to better performance and throughput of transaction with lower latency and lower transaction confirmation costs [239]. In addition, Hyperledger is not a token blockchain, and therefore the environment implementation is less complex.

Off-chain (IPFS): There are many limitations to using blockchain, such as the amount of data that can be stored in a single block, the potentially prohibitive cost of committing a transaction to a block, and the possible need to share information for verification. However, the data itself is not readable or confidential by all peers in the blockchain [240]. In addition, storing the data on-chain can cause the blockchain to grow excessively fast and negatively affect the network performance [144]. Thus, there is a need to utilise off-chain distributed storage systems such as MaidSafe, Storj and IPFS to offset these limitations. IPFS has been chosen to integrate with the architectural design because MaidSafe storage is still in alpha stage and Storj storage is a not a free service. IPFS is promised to offer a peer-to-peer distributed storage platform with high throughput [241]. In addition, data can be shared between organisations without editing or changing.

IPFS has an interface that can interact with the system through the Application Programming Interface (API) part of the framework architecture. IPFS splits files into chunks and stores them

in a distributed way to request and transfer them between nodes. When the data is stored off-chain, the generated content-addressed hash that identified the data is stored on-chain. When the data is needed, a peer can utilise the corresponding addressed hash to access the data. In other words, data is not part of the blockchain, but a fingerprint of the data is stored on-chain, reducing chain growth as content-addressed hashes are commonly smaller represented data. Generally, SHA256 stored on-chain requires 32 bytes for any file size that represents in IPFS. Each part of the model is discussed in more detail in the following sections.

5.2. Hyperledger Fabric Architecture

Traditional blockchain platforms such as Bitcoin and Ethereum are designed as order-execute architecture [235]. In other words, transactions are ordered first in the architecture, then executed sequentially on all peers in the same order, as demonstrated in Figure 5-2. This architecture process cycle limits the scalability of the blockchain as well as affecting the throughput and performance that can be achieved by the network. Moreover, smart contract in this architecture uses non-deterministic transaction, which means that all peers in the blockchain network hold the same state, operate the same transaction, and return the same outcome. This is because this architecture supports domain-specific languages such as Solidity to design smart contracts and does not support generic languages such as Go and Java.



Figure 5-2 Order-execute architecture

Hyperledger Fabric solves this problem in its innovation, as it uses architecture based on an execute-order-validation approach [235]. The architecture is divided into three phrases: the execution phase, where a smart contract (chaincode in Hyperledger Fabric) is created and run on one or more endorsers to execute transactions; the ordering phase, where transactions are grouped and ordered to submit them to the ordering service using the consensus algorithm; and the validation phrase, where network peers verify blocks received from the orderer before updating their ledger. Such an architecture allows transactions to be executed before ordering

it in the blockchain network to tolerating non-deterministic smart contracts. In later sections, Hyperledger Fabric's architecture will be addressed in more depth.

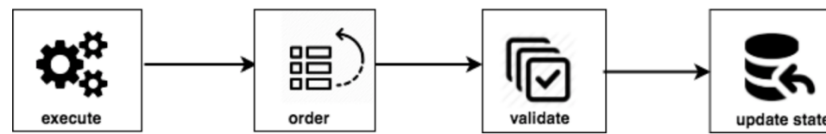


Figure 5-3 Execute-order-validate architecture

5.2.1. Architecture Components

Nodes in Hyperledger fabric have different roles and tasks assigned to them. They are categorised as client, peer, orderer, endorser, and committer.

- Client is a node that creates the transaction and can be any specific organisation's portal or application. The client can interact with the network by using the REST web service or Hyperledger Fabric SDK. It is responsible to invoke endorser peer to submit transaction proposals and receive endorsed transactions from endorser to broadcast it to the orderer peer.
- Peer is a node that receives invocation transactions from the client and maintains the distributed ledger. A peer may have one of several different special responsibilities: either orderer peer, endorser peer, or committer peer.
 - Orderer peer is a node from the ordering service, which provides a communication fabric channel for the network (i.e., it is considered a broadcasting service to clients and peers for messages containing transactions delivered as blocks).
 - Endorser peer is a peer that validates whether or not the transaction fulfils all the requirements after receiving it from the client application by checking the roles and certificate details of the requester. The endorser is responsible for executing the smart contract (called chaincode in Hyperledger Fabric), simulating the transaction's outcome, and returning the outcome to the client after appending its cryptographic signature. Therefore, the endorser is eligible to approve or disapprove the request. Note that an endorser peer can play a committer peer role as well. Figure 5-4 illustrates the difference between an endorser peer and a committer peer.
 - Committer peer is a peer that commits the transactions and updates the local ledger. All nodes are committer peers by default.

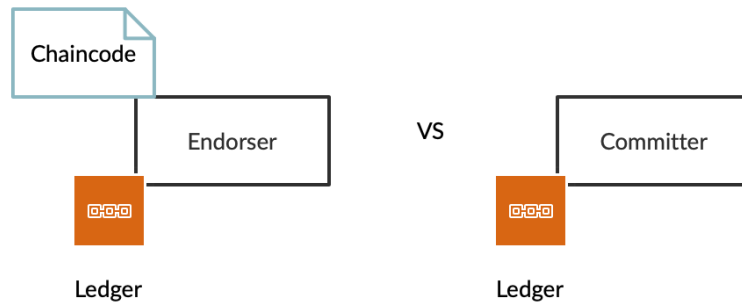


Figure 5-4 Endorser peer vs committer peer

Each peer’s ledger consists of two main components, world state database and blockchain, as shown in Figure 5-5. The former – also known as a blockchain ledger state – is initialised and maintained by chaincode to store the latest state values for each blockchain’s transaction log. Currently, there are two different types of blockchain ledger state database supported by Hyperledger Fabric: LevelDB and CouchDB [229]. LevelDB (called the key-value database) is embedded and built into peer nodes by default and stores key range queries, key queries, and composite key queries. CouchDB (called the client-server model), meanwhile, is an optional alternative that allows the issuing of rich queries rather than using the keys to allow operations to be performed by ID. The latter will be used in the implementation, since it records each chaincode’s transaction data on a particulate channel.

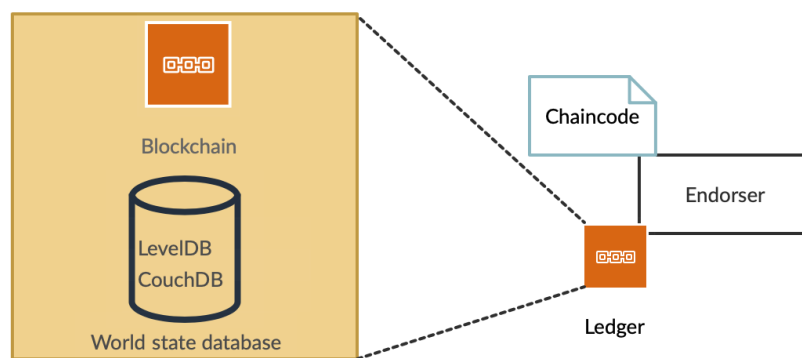


Figure 5-5 Inner components of peer’s ledger

Chaincode is another important component in the architecture is the same concept of smart contract. It is a program code that can be implemented in three languages: Node.js, GO, and JavaScript. Smart contract or chaincode contains business logic, and it manages the world state database (i.e. to interact with the peer’s ledger by external application, the chaincode must first be invoked). The chaincode executes the `get()` operation to read the world state and the

put () operation to state updates (see Figure 5-6). Therefore, chaincode must be instantiated on a channel as well as installed on an endorser peer.

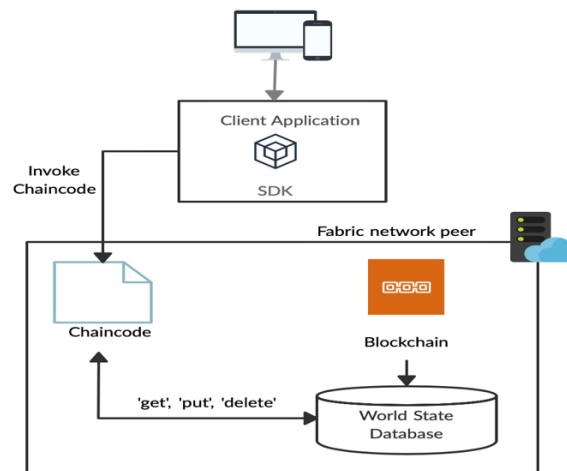


Figure 5-6 Hyperledger fabric chaincode

Every transaction on fabric network is accrued in a **channel**, which ensures privacy by maintaining an Access Control List (ACL). When the channel is created, the ACL is created at the same time. Each channel can be viewed and utilised only by the channel’s participating peers.

In a permissioned network such as Hyperledger Fabric, all peers that access information or create a new transaction are permitted. To control permissions, a **certificate authority (CA)** component is needed in order to issue identities to all participants in the network, including organisations, clients, and nodes. Hyperledger Fabric’s CA – named Fabric-CA – plays an important role in operating authorisation and managing identity. Fabric-CA utilises a public key infrastructure to generate x.509 certificates to each member of the organisation. One root certificate is generated for each organisation. For one-time transaction clients, in addition, Fabric-CA can generate temporary certificates.

5.2.2. Creating the Fabric Network

As shown in Figure 5-7, the first step to creating a network N is to develop an orderer peer. In the blockchain network, there is one orderer in the ordering service O1 that follows the network configuration policy. Fabric certificate authority CA1 generates certificates for organisation R1 to allow it to interact with the network and to be a part of it. In addition, these certificates are used by the organisation’s client applications to authenticate transaction proposals and by the organisation’s endorsers to issue digital signatures to represent their transaction results.

Each organisation can utilise Fabric-CA, which is provided by default in Hyperledger Fabric, or can choose its own certificate authority company.

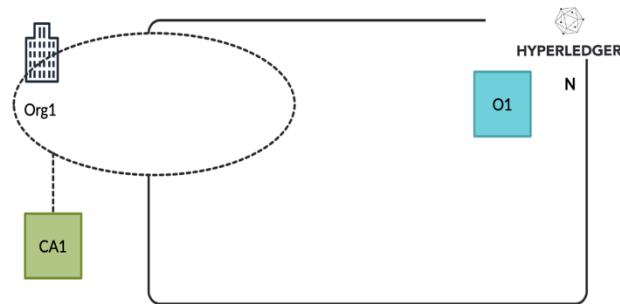


Figure 5-7 Creating the fabric network

Since Hyperledger Fabric is a permissioned blockchain, it is necessary to create an administrator, as shown in Figure 5-8. Org1 is initialised as an organisation that follows the administrative right policies in the network configuration. Org1 has permission to add another administrator, Org2. In this case, both Org1 and Org2 follow the same right policies in the network configuration. Org 2 must have certificates for its users to able to use the network by adding CA2.

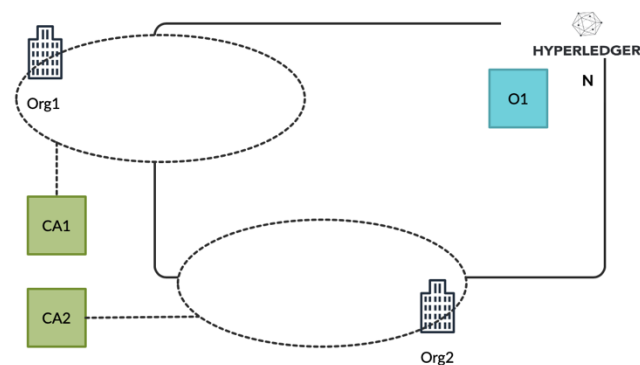


Figure 5-8 Adding organisations as administrators

After this, the next step is defining a consortium. Organisations must be part of a consortium before adding them to a channel. Between Org1 and Org2, a new consortium is formed, and they must agree to the network's governing policies. Creating a consortium between different organisations is mandatory for every transaction. Then a channel must be created and governed by channel configuration to allow consortium organisations to share the network infrastructure and communicate privately (see Figure 5-9).

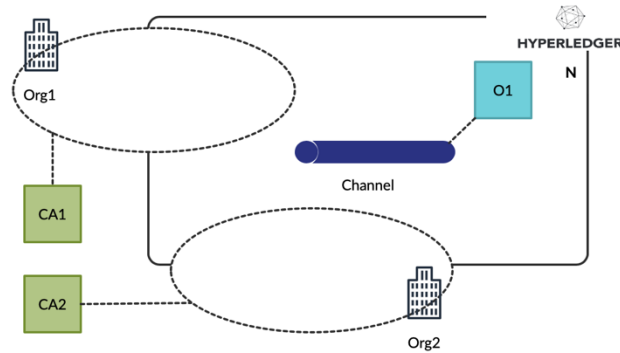


Figure 5-9 Creating a channel

When peers are joined to the channel, there can be several peers owned by different organisations, as shown in Figure 5-10. P1 is a peer node that becomes part of the channel C1. P1, owned by Org1, maintains a ledger, and the channel C1 also has a copy of it. Physically, the ledger is held in P1, but logically it is stored on C1. In addition, P4, owned by Org2, is added to the channel C1 and has the same copy of the ledger.

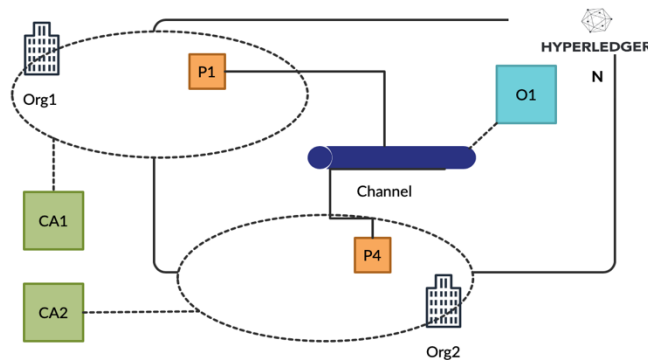


Figure 5-10 Defining peers

The next step is installing client applications and smart contracts (chaincodes), as indicated in Figure 5-11. The client application, which is placed out the fabric network, is responsible for interacting with the ledger in a peer by generating transaction proposals. A smart contract in P1 assists client application A1 to access the ledger through P1. In other words, client application A1, which is owned by Org1, cannot access the ledger in P1 directly. It must invoke its smart contract first. The smart contract must be installed on every node in a network and instantiated at least in a node on an organisation. It must be installed on P1 as well as on the channel C1 in order to permit other components connected to C1 to be aware of the smart contract.

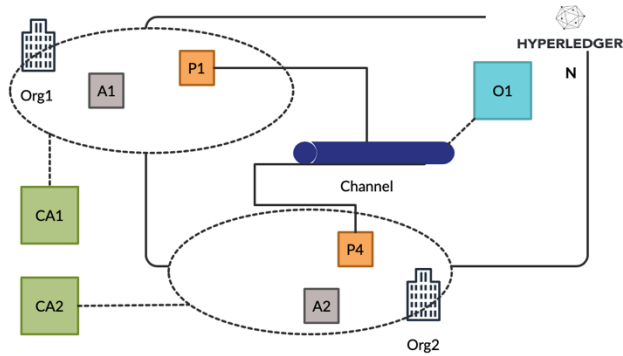


Figure 5-11 Adding fabric client applications and chaincode (smart contract)

To simplify the network that has been developed, as can be seen in Figure 5-11, there are two organisations connected to the fabric network through client applications. Each organisation has a peer node that is connected to a single ordering service on a channel. Connecting to a single channel means that there is only one logical ledger in the network. Thus, P1 and P4 have identical copies of the smart contract and the ledger. The network can be expanded by defining more organisations and peers that follow new consortiums, as well as creating more channels with channel configurations. Each channel in the same network retains separate policies. Ordering services on the same network can control multiple channels and can be multiple ordering services. More connected channels lead to the development of more smart contracts and logical ledgers.

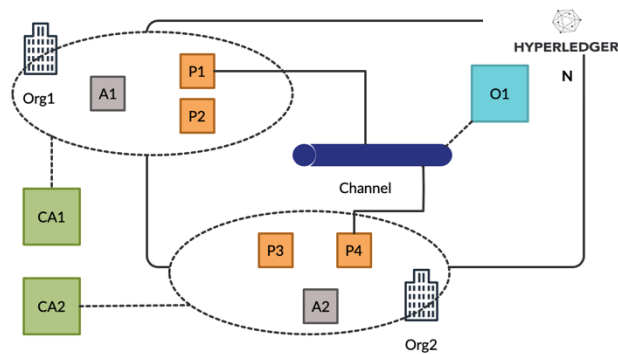


Figure 5-12 Completing the fabric network

5.2.3. Transaction Flow in the Fabric Network

As was explained earlier, the peers on Hyperledger Fabric host ledgers that can be invoked and queried by the peers' applications via chaincodes. An invoked transaction requires updating the ledgers, which is a different process to a query transaction. Updating the ledger cannot be complete by a solo node; it needs the other peers on the network to approve the changes before

applying it to the peers' ledgers. Therefore, updating a transaction requires additional steps to process which called consensus more than a query transaction process which needs two phase steps to be done.

- **Proposal phase:** This is the first step of transaction flow, which occurs between a client application and endorsing peers. The main aim of this phase is to generate a proposal and ask endorsing nodes to validate to the results thereof. When the application supported by SDK generates the transaction proposal, it sends it to the required set of chosen endorsing nodes, which is defined by endorsement policy. Each of these endorsers must independently generate a response to the transaction proposal by executing their chaincodes using the proposal as input argument. Executing the chaincode produces a transaction result including a read set, and write set (i.e. key pairs as an asset to update or create) and response value. The endorser does not change its local ledger at this stage. What it does do is to sign the proposal using its private key and return it back to the application. The first process of transaction workflow is ended when the application receives a number of signed transaction proposal responses.
- **Ordering phase:** After the application receives proposal responses containing a set of values with endorsers' signatures, it starts to verify the signatures and compares the results to ascertain whether they are same or not. If the proposal transaction invokes a query from the chaincodes, the application would determine the query results and would not continue to send the transaction to the ordering service (i.e. the invoke transaction process would end at this phase). If the proposal transaction asks ledgers to be updated, then the application sends the transaction to the ordering service after determining whether or not the endorsement policy has been completed. The ordering service simultaneously receives several transactions from different applications and channels in the network. If there are many orderers on the network, they collaborate together to form the ordering service. The transaction includes read/write sets attached with endorser signature, plus the channel ID. When the ordering service receives transactions, it arranges them chronologically by channel and packages them to generate blocks. These blocks will later be attached to the blockchain. Depending on what governs the configuration parameters, they determine the size and number of transactions that can be in a block, and the duration time holds up after arriving the first transaction before cutting a block for additional transactions. Note that the size of the block cannot exceed the maximum number of bytes. The blocks are then broadcasted

to all peers per channel. Therefore, this phase depends on the ordering service, which is responsible for collecting authorised transaction updates and packaging them in blocks after ordering them to be ready for broadcasting.

- **Validation and committing phase:** This stage is the last phase of the transaction flow, where the blocks are distributed and validated from the orderer to the peers, and the peers then commit them to their ledgers. When each peer in the channel receives a transaction as a block, the peer starts to validate it independently by checking if the endorsements match the chaincode's endorsement policy to ensure it has been validated by the required endorsers. Because the orderer packages and sends every authorised transaction to peers without making any judgement on the content except checking the governs configuration transactions, as discussed previously, peers must check whether the transaction is valid or not. If it is invalid, the peers will tag it as an invalid block without updating their ledgers' states. If it is valid, each peer on the channel appends the transaction block to its blockchain as well as updating its current state database using the write sets. It should be noted that validation in this phase is different than in the previous phase, where the client application receives transaction proposal responses and checks the endorsement policy before sending the proposal transaction. Thus, the peer in the validation phase is still able to dismiss the transaction in case the client application sends wrong transaction without checking the endorsement policy.

In addition, this phase does not require execution of chaincodes for validation as in the first phase, which is an important feature of Hyperledger Fabric since chaincodes are applied on a required organisation's endorsers and not throughout the network. In this way, the chaincode can be confidential to endorsers. However, the results of the chaincodes that form the transaction proposal responses are broadcasted to the channel's peers whether they are endorsers or committers, which leads to increased confidentiality and scalability in the network. Finally, in this phase, each peer has to emit an event to the client application to notify it that the invoking transaction has been added to the blockchain immutably, as well as notifying whether the block was validated or not.

The overall transaction flow of Hyperledger Fabric is shown in Figure 5-13.

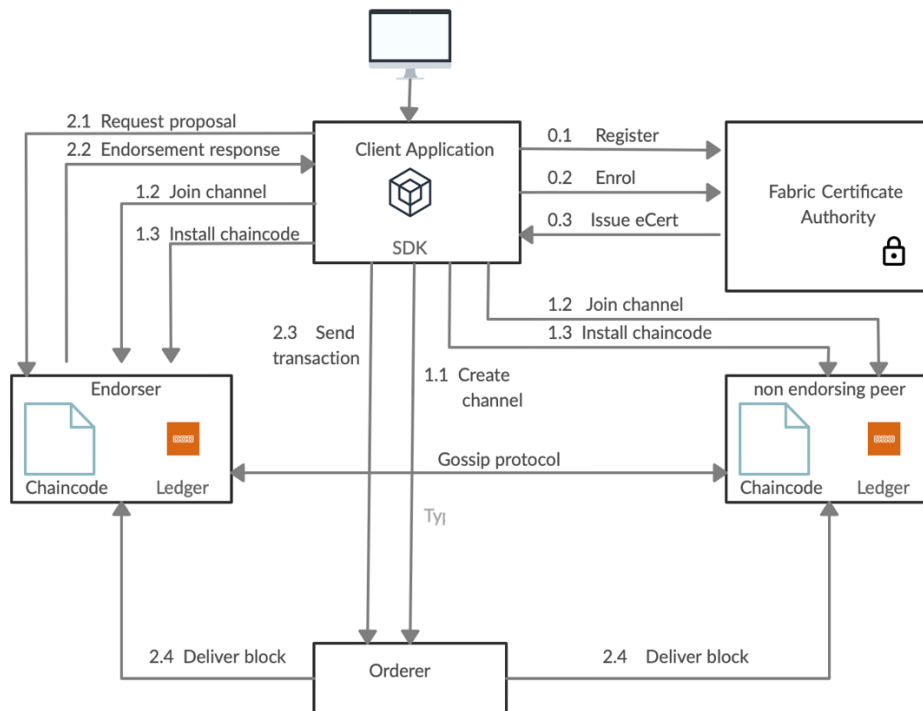


Figure 5-13 Hyperledger Fabric transaction flow

To configure Hyperledger Fabric's processes, the Docker platform has been used to run range of nodes. From the client, a transaction proposal can be created via a function from Hyperledger Fabric SDK, which must in turn be signed from the Fabric Certificate Authority (FCA). The FCA only has to be accessed when registering new certificates to allow a peer to join the network. After that, an endorser who hosts the ledger and chaincode receives transaction proposals to run the code and signs the proposal to return it to the client. The client then sends the transaction to the ordering service responsible for the ordering of blocks by running the consensus algorithm to validate a block and distribute it to the peers to update their ledgers.

Next, IPFS as a decentralised off-chain storage component will be highlighted and discussed in terms of how it can be used in the model.

5.3. Off-chain Storage: IPFS

Off-chain storage is another important part of the model that refers to any storage that is located outside of blockchain. Since the distributed ledger in the blockchain has limitations in terms of how much data can be stored, large non-transactional data that leads to undesirable chain growth can be stored off-chain. Using off-chain storage also prevents degradation of

Hyperledger Fabric's performance in handling large data in the blockchain [240]. In practice, smart campus data is usually too large to store in a distributed ledger, and it is efficient to store it off-chain.

Since the goal is to build distributed architecture, distributed off-chain storage has been used: specifically, IPFS. Large data, such as images, are stored in IPFS, and a small fraction of this data is stored in the blockchain ledger. Therefore, ledgers and chaincode functions are not affected by big data. In other words, the only thing that is recorded on-chain is a pointer or the address of the individual location field that is in IPFS.

IPFS is a peer-to-peer protocol network for distribution and storage of data in a decentralised file system. IPFS breaks files of a size greater than 256KB into several blocks of 256KB, then refers to them using hash pointers. IPFS has many properties that are advantageous for the use cases, such as the following:

- **No file type restrictions:** IPFS does not restrict on file formats, and all files are treated the same way as simple text format (i.e. it is unstructured data storage). For the architecture, this feature is vital, since smart campus data has multiple types and is spread widely.
- **Easy sharing of data:** Since IPFS is distributed storage, its protocol provides a hash value for each file or data saved. This sharing protocol allows different peers and nodes in the network to share files easily.
- **Tamper proof:** Data in IPFS can be found by content addressed not location addressed such as HTTP. To determine the file location in content-based storage, a content identifier is needed. IPFS creates a content identifier based on the data cryptographic hash (i.e. it is mostly a fingerprint of the data). In this case, it is extremely difficult to receive the same cryptographic hash when the file content is changed. Therefore, file content in IPFS cannot be overwritten, which ensure data integrity and a self-certifying storage system.

Combining an off-chain storage system with a blockchain technique decreases some concerns in the architecture such as the following:

- As data is stored off-chain and only its fingerprint hash is saved on-chain, this leads to a risk of data availability. In other words, data availability considered as an off-chain storage danger when data is not part of the blockchain. However, IPFS is a decentralised

system and promises to ensure data availability by distributing and hosting data among different peers. Data can be fetched from any nearest peer in the network instead of fetching it from traditional client-server architecture.

- Data can only be stored by peers that have permission to access data and does not need to be hosted by all peers in the network. For example, student data is only required to be held by IPFS peers that relate to the framework, and the lecturer can attach data to which only he or she has authorised access to the off-chain storage system. Therefore, there is the benefit of decreasing the amount of data that needs to be store by each peer.
- Computation of off-chain storage heavily decreases the redundant computation needed to perform consensus in the network. Since on-chain holds only a fingerprint of the data that is stored off-chain, blockchain can process queries with less computation as most computation is done off-chain.

5.3.1. Transaction Flow for Integration of Hyperledger Fabric Blockchain and IPFS

Section 5.2.3 discussed a transaction flow for the Hyperledger Fabric network. The data in that case after the validation process will be stored in the peer's ledger itself or on-chain, which will cause the blockchain to grow excessively fast and negatively affect the network performance as discussed earlier. Therefore, there is a need to utilise IPFS as an off-chain distributed storage system and integrate it with the framework. Thus, data that is stored off-chain is non-transaction, which is usually too large in size to handle directly in blockchain.

As can be seen in Figure 5-14, IPFS has been added to the previous transaction flow chart to provide off-chain storage.

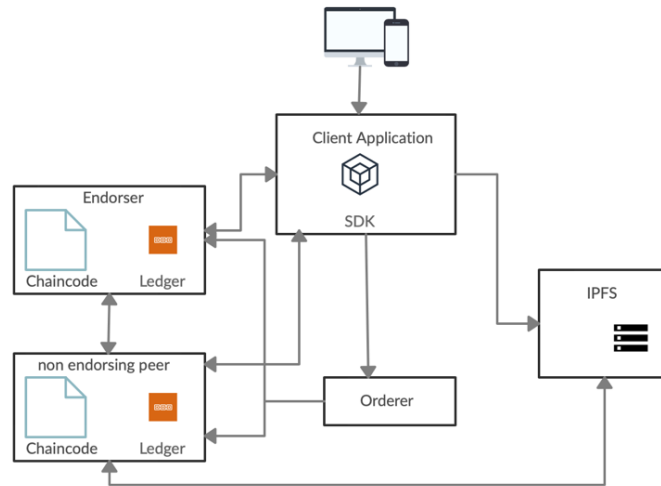


Figure 5-14 Transaction flow for integration of Hyperledger Fabric and IPFS

The main components of the above figure and their roles are presented as follows:

- **User:** The user is the actor who accesses or/and adds a data on the IPFS component. Depend on the user rights on the network, they can perform operations such as get, cat, or add on data. The operation command can be as follows:

```
$ ipfs add mydoc.doc
```

- **Client application:** The client application is responsible for interacting with the backend, including on-chain and off-chain storages. To access the data, the client application retrieves the data hash address from the Fabric Hyperledger blockchain after executing the chaincode. They then invoke IPFS to obtain the data information. While storing the data, the client application calls IPFS to store the data. Then it invokes the blockchain with the data hash address to execute and verify the transaction. In the implementation, Java Script API has been used.
- **REST Server:** This component is necessary to allow the client application to connect to the Hyperledger Fabric network as well as the IPFS network by utilising a gateway and packing to implement the required methods for the front end side.
- **Hyperledger Fabric network:** The fabric blockchain is explained in terms of its components and transaction flow in more detail in section 5.2.
- **Data:** In the approach, data can be described as any object that is stored in IPFS. The data can be any type of file, such as document, video, audio, image, etc. There is no size limitation on IPFS.
- **IPFS network:** The IPFS network is where the data is stored off-chain and its hash is generated to be stored on the blockchain. Since the system handles sensitive data,

private IPFS is used in the implementation, in which all peers on the network know each other. All nodes in the private IPFS share a secret key known as the ‘swarm key’ to help them identify each other and allow them to respond only to the nodes inside the network.

In Figure 5-14, two scenarios can occur in the proposed solution: uploading data and accessing data. Each will be discussed as follows:

- 1 Uploading data:** When an authorised user requests to upload a file to the network utilising add command, the client application invokes the private IPFS client to add the file in IPFS. IPFS in turn generates and returns the data hash. The IPFS client will invoke Hyperledger Fabric SDK through the application client to pass on the data hash and the file metadata, such as the user who added the file and the date and time of upload in order to store them on the blockchain. The Hyperledger Fabric SDK then will call the chaincode to update the blockchain. The blockchain network runs its process to verify and add the transaction into the ledger. When the ledger is updated, the user will receive an alert to say that the update has been successful. If it is not successful, the user will receive an alert of a failed process with the option to cancel or continue.
- 2 Accessing data:** To access or fetch the data stored in IPFS, the client application invokes Hyperledger Fabric SDK in order to invoke and run the chaincode to get the corresponding data hash that is stored on the blockchain. The chaincode returns the data hash as well as storing access metadata such as the accessed hash data and the node public address that accessed the data and timestamp to the ledger. Then the client application invokes the IPFS client and passes the corresponding addressed hash to read the data from the private IPFS network. IPFS then returns the file content to the user.

5.4. Summary

This chapter presented the design of the smart education environment framework. The framework used Hyperledger Fabric blockchain to decentralise the system, and it delivers decentralisation, data integrity, and security. Since blockchain generally has scalability concerns when such an environment has massive amounts of data that can lead to excessively fast growth of blocks and negatively affect network performance, and to minimise the cost of transaction commitment, decentralised off-chain (i.e. IPFS) storage was used to store the data

itself, and only transactions were executed on-chain. This design framework will be implemented in different case scenarios in the following sections.

Chapter 6 A Smart Campus Framework Architecture Implementation

After designing the smart education architectural framework using Hyperledger Fabric blockchain and interplanetary file system (IPFS) as a decentralised off-chain storage to store and retrieve data, this chapter presents the use of previous architectural framework by implementing different case scenarios in a smart campus – in particular, a smart educational environment as a proof of concept to continue answering the third research question. After that, the chapter will examine the fourth research question: ‘What are the implications of the applicability of a comprehensive framework for a smart campus in the future?’

Deployment of Hyperledger Fabric network for the case scenarios is described in section 6.1. For implementation, Hyperledger Fabric version 2.3.2 was used as it was the latest version at the time this thesis was written. Section 6.2 illustrates the university library management system as the first case scenario using the proposed architectural framework. Section 6.3 discusses the second case scenario of generating a student credential report, including storing student records and generating and sharing a student report. Section 6.4 presents the testing environment used to evaluate the framework. The source code for the implementations and the testing can be found at GitHub <http://github.com/Manal979/Project> , see Appendix. Section 6.5 presents the applicability of the comprehensive framework for a smart campus in a broader context. Finally, section 6.6 provides overall results discussion.

6.1. Deploy Hyperledger Fabric Network

Setting up the network required installation of a series of Fabric prerequisites, including Git, cURL, NPM v8.1.0, NodeJS v16.12.0 and Go programming language. Then, the Fabric platform-specific configuration and binaries files were installed, which consists of the most important directories, /config and /bin. Lastly, the Hyperledger Fabric docker v20.10.7 and Docker compose v1.17.1 were installed. Docker was used to provide containers for each

component, such as peers, orders and CA. Instead to use full physical or virtual machines, container technology provides more lightweight feature for the implementations and allows testing configuration files much faster.

Table 6-1 Environment configuration

Component	Description
CPU	Intel Core i7
Memory	12 GB
Operating System	Ubuntu Linux
Hyperledger Fabric	v2.3.2
NPM	v8.1.0
Node	v16.12.0
Docker	v20.10.7
Docker Compose	v1.17.1
CLI Tool	Angular
DBMS	Couch DB

Our building Fabric network contains two organisations, and each has two peers, which are the network participants or actors. All components connect to one channel and deploys a Raft consensus algorithm for three orderers, as can be seen in figure 6-1 YAML definition of bootstrapping the network. All certificates are managed by the CA authorities created for both organisations.

```

fabric:
  cas:
    - "ca.org1"
    - "ca.org2"
  peer:
    - "peer0.org1"
    - "peer1.org1"
    - "peer0.org2"
    - "peer1.org2"
  orderers:
    - "orderer1.orderer"
    - "orderer2.orderer"
    - "orderer3.orderer"
  settings:
    ca:
      FABRIC_LOGGING-SPEC: INFO
    peer:
      FABRIC_LOGGING-SPEC: INFO
    orderer:
      FABRIC_LOGGING-SPEC: INFO

```

```
netname: "project"
```

Figure 6-1 Definition of bootstrapping the blockchain network

Since setting up the network is reliant on multiple configuration files, configuration scripts and files were modified to meet the needs, as per the following:

- **Configtx.yaml**: This file has configuration information and channel transaction files for developing the genesis block. The file consists of several sections:
 - **Organisations**: This section defines the individual organisations with their identity details, which are referenced in the configuration file. The details include the organisation's name; Membership Service Provider (MSP) ID, which is responsible for handling all the cryptographic operations such as issuing, signing, verifying and chaining; MSPDir, which is a directory that contain the organisation's crypto materials and MSP configuration; and Anchor peers, which are used to keep data sync between organisations' peers in the gossip protocol by specifying the host and port of them; Finally, the orderers need to be defined. In the implementation, there are three orderers.
 - **Orderer**: This section defines all the orderer parameters. These parameters include the orderer consensus algorithm; the port and host addresses; BatchTimeout, which is the time that the orderer needs to wait before creating a batch; MaxMessageCount which means the maximum size of a block; AbsoluteMaxBytes which is the absolute maximum number of bytes to permit in a block; and preferredMaxBytes, which is the preferred maximum size of bytes for a block.
 - **Capabilities**: This section is used in the case that the network nodes running different versions of Fabric code to avoid any impacts. Capabilities can be deployed in three different places, including channel, where both peer and orderer can communicate; Orderer, where capabilities can be applied in the orderer group; and application, where deployment to peers only.
 - **Application**: This section uses the application defaults, which is referred in the genesis block.
 - **Profiles**: This section is considered a vital part in the Configtx.yaml file since it combines all the previous configurations and describes the structure of the network. It contains of two parts: the first part is related to a channel configuration and the second part is related to genesis block configuration.

- `Crypto-config.yaml`: This file has the generated certificates and key information for the organisations and their components. When the cryptographic generator tool (cryptogen tool) is used to generate key material and digital certificates for the network, this file is required.
- `Fabric-ca-config.yaml`: Here, the CA server provided by Fabric was used to issue certificates for the entities, however, in real life, intermediate CA provided by any trusted enterprise CA should be used. This file contains the CA parameters including private and public keys that used for issuing and signing certificates.
- `Docker-compose.yaml`: This file has the docker containers configurations for the network entities such as orderers, peers, cli and CouchDB. These entities' configurations include their ports, domains, addresses, paths, and other essential parameters that are important to set up the network correctly. In addition, this file is responsible for setting up CA containers.

In the next section, the first case scenario will be presented in more detail.

6.2. Library Management System Case Scenario

Because the area of library management systems is slow to embrace new technologies [116], a library management system case scenario was used to demonstrate the smart educational architectural framework. Library functionality has not changed over the decades. There is still in place a framework that saves enormous amounts of information and materials. Library systems traditionally need physical storage space and maintain their availability by providing physical accessibility to users. However, managing and storing such resources as articles, books and magazines can be very costly, and the facilities may need to collaborate with other organisations or operate with several branches. Although computers and digital devices provide extremely high information storage densities and do not suffer from the problem of a traditional library, the complexity of the library is less affected [115], [117]. The issues of records integrity and storage effectiveness are present in the digital library since the information infrastructure relies on centralised structure that may suffer from such attacks as single point of failure or DDoS, as explained in more detailed in Section 3.2. In addition, the current centralised library systems may suffer from unauthorised access to materials.

The blockchain technology that was presented in the literature review, particularly the Hyperledger Fabric blockchain, fits well with library management systems. It can eliminate a central authority by using a distributed ledger to build a decentralised system, therefore providing more security and efficiency for operating and controlling communication among all participating nodes to the library recourses and archives. Moreover, blockchain technology uses cryptographic mechanisms, as explained in Section 2.3.2. This can be tamper-resistant, with a potential use for scientific publications and journal articles that are stored in the library system to ensure that the materials have not been changed or altered. Because data are stored in and retrieved from the distributed ledger, the library systems can ensure data integrity and prevent a single point of failure. There are several studies that focus on emerging blockchain technology to different library management systems to increase their security, as was addressed in Section 2.4.3. However, there is no existing implementation for this type of platform.

This section’s adoption of the Hyperledger Fabric and IPFS to the library management system has been used to extend the current theory studies. In this framework, multiple libraries can be joined to the network and books can be listed. Students who are enrolled in the network can see the metadata of books, and seamlessly check the availability of a book, borrow it, extend their loan, or return it.

The implementation of this uses the Hyperledger Fabric blockchain (that was implemented in section 6.1) to store a book’s International Standard Book Number (ISBN), which is a unique identifier, and its IPFS hashes. In addition, the implementation uses a private IPFS network to store the book’s metadata, such as the book title, the author’s name, the publisher, and the publication year. Figure 6-2 shows the flow diagram of the proposed solution.

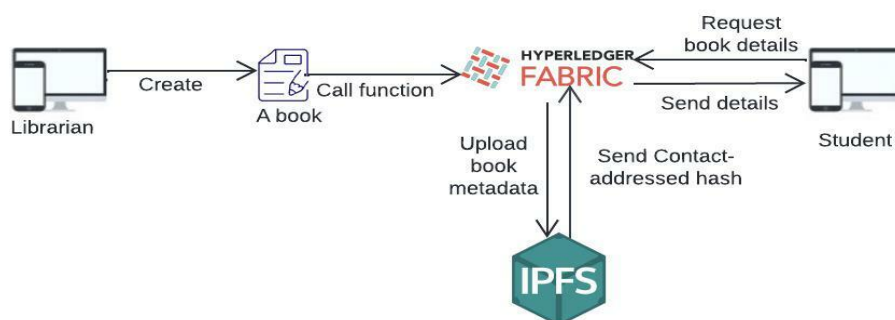


Figure 6-2 library management system case scenario

The diagram shows the architecture of the library management system based on off-chain storage. It is fully decentralised architecture, guaranteeing transparency and security as well as

eliminating the trusted third party. There are two roles in the system, librarians and students, which are described as follows:

- **Librarians:** The librarians from the library organisation have rights to read and write to the ledger. They are responsible for calling and executing the chaincode to store the books' information on the blockchain.
- **Students:** Students have rights to read the ledger without writing or updating it. Students can invoke the chaincode to access a book's information, such as author and publisher name, book content and number of copies. In addition, students can borrow a book, check the availability, extend a book or return it.

In figure 6-3, a sequence diagram explains the interaction between system components and the way the librarian stores the book's data on the system.

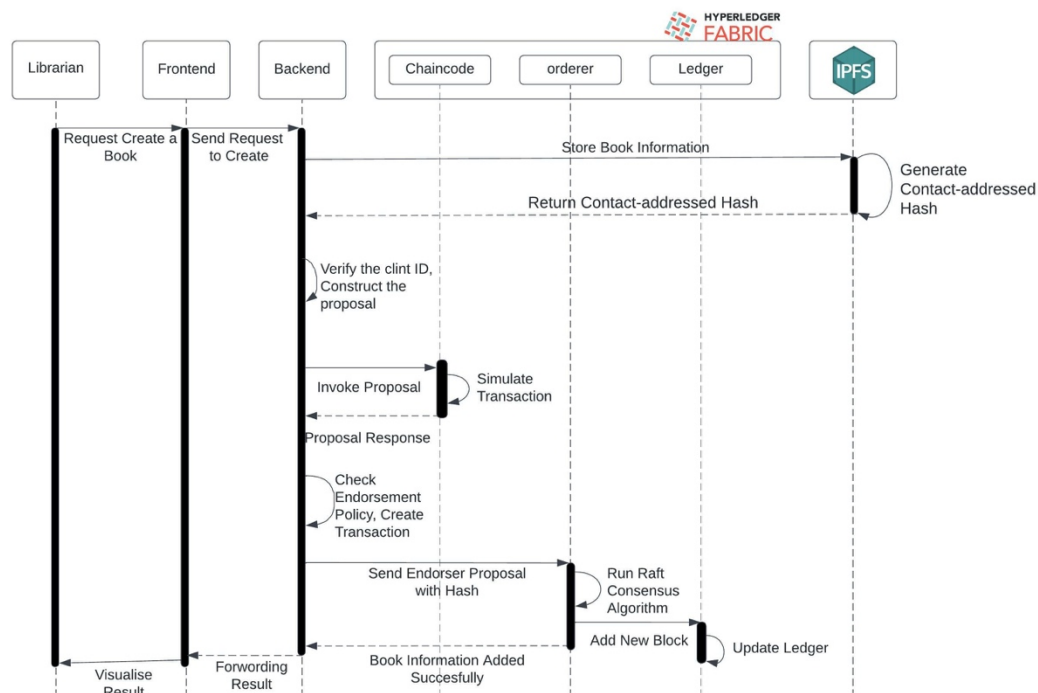


Figure 6-3 A sequence diagram for storing book information

First, the librarian requests the creation of a book in the application layer by giving all the required information, such as book ISBN, title, author's name, number of copies, publication year, and publisher name. The information is sent to the backend through the application layer. The backend segregates the data and calls IPFS to store the book title, author's name, publication year, and the publisher's name, while the book's ISBN and the number of copies will be stored later in the ledger because the ISBN will be used as a key in the ledger to help

fetch the book information from the IPFS. The number of copies is a changing variable that cannot be stored on IPFS, which is more suitable for storing static data. IPFS stores the data by distributing it across the network and returns the data hash to the backend layer. The data hash can be used as a fingerprint of the data to be stored on-chain and used later by other peers to access the data location on IPFS. The backend layer creates the transactions that contains the ISBN, the number of copies, the data hash and the data timestamp, and signs it digitally by utilising the user certificate authority, which is generated by Fabric-CA. Then the transaction is sent to the Fabric Hyperledger blockchain to be stored in the ledger. The ‘endorser peer’ in the blockchain is responsible for executing the smart contract, simulating the transaction’s outcome – which consists of the world state and the read-write set – and returning the outcome to the backend after appending its cryptographic signature. The backend collects all the endorser peers’ transaction responses and verifies the validity of the endorsers’ signatures. After that, the backend creates the proposal and sends it to the ordering service to run the Raft consensus algorithm, verify the transaction, and create a block to be broadcast to all participants in the network. The peers, in turn, check the validity of the transaction block by verifying the endorser’s policies and determining whether the read set changed or not. After meeting all the validation requirements, the transaction block is then marked either valid or invalid, allowing the peers to update their ledger and append the transaction block in the blockchain. The backend is then notified that the block has been appended successfully. Finally, the application layer forwards the notification to the client and visualises the results.

In Figure 6-4. a sequence diagram presents the interaction between the system components that enable the student to search for a desire book.

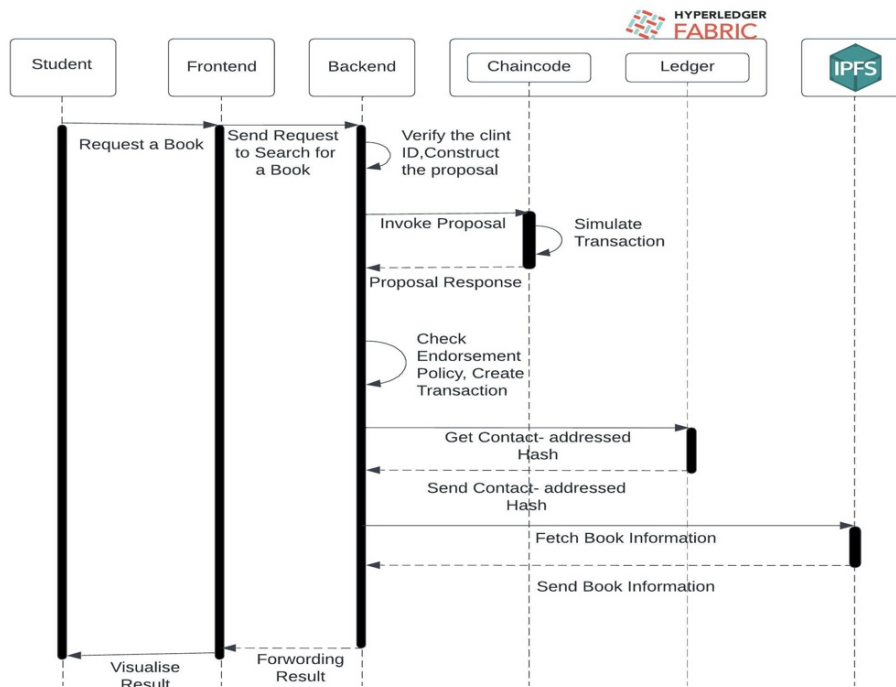


Figure 6-4 A sequence diagram for searching for a desire book

When a student requests to search for a book, they need to enter the ISBN for the book. The application layer will call the backend and pass the ISBN to it. The backend in turn will create a transaction proposal, which contains the ISBN and signs it digitally, utilising the user certificate authority, which is generated by Fabric-CA, and calling the blockchain endorsers through Fabric SDK to execute the query function in the chaincode. The endorser peer simulates the transaction's outcome, which consists of the world state and the read set, and returns the outcome to the backend after appending its cryptographic signature. The backend collects all the endorser peer's transaction responses and verifies the validity of the endorser's signatures. In terms of reading a value, the value is the data hash that was generated by IPFS from the ledger; there is no need to call the ordering service because generating a block is not required. After the backend layer has the content-addressed hash, it will be sent to IPFS to fetch the book's metadata. The book information will then be displayed to the student, and they can borrow a book, check its availability, renew a loan, etc.

The next sections will describe the other requirements for implementation, including the scenario's chaincode, API, frontend, and dashboard.

6.2.1. Smart Contract (Chaincode)

There are several transactions that can be performed in the network by the participants:

Create book: The chaincode function `createBook` is the method to insert the book information into the ledger. It accepts parameters such as `ctx`, `ISBN`, `copies`, `data hash`, `additional data`, where `ctx` is transaction context to help provide access to the API, and `additional data` is the JSON metadata for recreating the IPFS data. The function `ctx.stub.putState` is used to create the state in the ledger. It accepts a key and value as parameter, where the key is the ISBN and the value is the JSON string of the book data – see figure 6- 3.

```
async createBook(ctx, isbn, copies, datahash, additionalData) {
  console.info('===== START : Create Book =====');
  additionalData = JSON.parse(additionalData);
  const book = { isbn, docType: 'book', copies, DataHash : datahash,
  additionalData:additionalData
  };

  await ctx.stub.putState(isbn, Buffer.from (JSON.stringify(book)));
  console.info('===== END : Create Book =====');
}
```

Figure 6-3 A code snippet of the create book function in the smart contract

Query all books: this function is responsible for querying the book information from the ledger. As can be seen in figure 6-4, the function accepts parameters such as `ctx` and `query`, where `query` is a JSON object with a selector or filtering parameters. This function returns the array of the book as a `βresponse`: `[{_id, bookname, isbn, numberofcopies, yearofpublishing, author, publisher, timestamp, cid}]`– see figure 6- 4.

```
async queryAllBook(ctx, query) {
  console.info('===== START : QUERY ALL Books TRANSACTIONS =====');
  let iterator = await ctx.stub.getQueryResult(query);
  const allResults = [];
  while (true) {
    const res = await iterator.next();
    if (res.value && res.value.value.toString()) {
      console.log(res.value.value.toString('utf8'));
      const Key = res.value.key;
      let Record;

      try {
        Record = JSON.parse(res.value.value.toString('utf8'));
      } catch (err) {
        console.log(err);
        Record = res.value.value.toString('utf8');
      }

      allResults.push({ Key, Record });
    }
  }
  if (res.done) {
```



```

        console.log('end of data');
        await iterator.close();
        console.info(allResults);
        return JSON.stringify(allResults);
    }
}
console.info('===== END : QUERY ALL Books TRANSACTIONS=====');
}

```

Figure 6-4 A code snippet of the query all books function in the smart contract

Query single book: This function retrieves the book data from the ledger by passing the ISBN. It uses the `getState` method to retrieve the state from the ledger. This function returns a book data object that contains `{_id, bookname,isbn, numberofcopies, yearofpublishing, author,publisher, timestamp, cid}` – see figure 6- 5.

```

async queryBook(ctx, isbn) {
    console.info('===== START : QUERY A Book =====');
    const bookAsBytes = await ctx.stub.getState(isbn); // get the book from
    chaincode state
    if (!bookAsBytes || bookAsBytes.length === 0) {
        throw new Error(`${isbn} does not exist`);
    }
    console.log(bookAsBytes.toString());
    console.info('===== END : QUERY Book END =====');
    return bookAsBytes.toString();
}

```

Figure 6-5 A code snippet of the query a book function in the smart contract

Issue a book: This method issues a book to the student through the ledger. The function accepts the inputs: ISBN; ‘issuedTo’ as a user ID; and ‘issuedOn’ as timestamp. Then it creates an issuance entry in the ledger and stores the transaction. Internally within the function, the availability of the book would be reduced by one – see figure 6-6.

```

async issueBook(ctx, isbn, issuedTo, issuedOn {
    console.info('===== START : ISSUE Book =====');
    const book = {
        isbn,docType: 'Tbook', issuedTo: issuedTo, issuedOn: issuedOn,
        status1:"Issued" };
    const bookAsBytes = await ctx.stub.getState(isbn); // get the book from
    chaincode state
    if (!bookAsBytes || bookAsBytes.length === 0) {
        throw new Error(`${isbn} does not exist`);
    }
    const bookOld = JSON.parse(bookAsBytes.toString());
    bookOld.copies = bookOld.copies - 1;
    await ctx.stub.putState(isbn, Buffer.from(JSON.stringify(bookOld)));
    await ctx.stub.putState("t" + isbn, Buffer.from(JSON.stringify(book)));
    console.info('===== END : ISSUE Book =====');
}

```

Figure 6-6 A code snippet of the issue a book function in the smart contract

Return a book: This method returns a book back to the library. It is primarily invoked from the student’s side. The function accepts the inputs: ISBN; ‘returnedBy’ as userId; and ‘returnedOn’ as a timestamp. Then it creates a ‘book returned’ entry in the ledger and stores the transaction. In addition, the availability of the book would be incremented by one – see figure 6-7.

```

async returnedBook(ctx, isbn, returnedBy, returnedOn, status1 {
  console.info('===== START : ISSUE Book =====');
  const book = {
    isbn, docType: 'Tbook', returnedBy: returnedBy, returnedOn:
    returnedOn, status1: "Returned" };
  const bookAsBytes = await ctx.stub.getState(isbn); // get the book from
  chaincode state
  if (!bookAsBytes || bookAsBytes.length === 0) {
    throw new Error(`${isbn} does not exist`);
  }
  const bookOld = JSON.parse(bookAsBytes.toString());
  bookOld.copies = bookOld.copies + 1;
  await ctx.stub.putState(isbn, Buffer.from(JSON.stringify(bookOld)));
  await ctx.stub.putState("t" + isbn + returnedBy,
  Buffer.from(JSON.stringify(book)));
  console.info('===== END : Return Book =====');
  return { "status" : false };
}

```

Figure 6-7 A code snippet of the return a book function in the smart contract

Delete a book: This method deletes a book from the ledger. The function accepts ISBN as an input and removes the entry from the ledger. One blockchain feature is immutability, in which the data cannot be changed while it is stored. In the delete a book function, what actually happens is that the book is marked as ‘true’ so it does not appear in the system, while in the ledger’s history, it still exists – see figure 6-8

```

async deleteBook(ctx, isbn) {
  console.info('===== START : Delete a Book =====');
  const bookAsBytes = await ctx.stub.getState(isbn); // get the book from
  chaincode state
  if (!bookAsBytes || bookAsBytes.length === 0) {
    throw new Error(`${isbn} does not exist`); }
  const book = JSON.parse(bookAsBytes.toString());
  book.delete1 = 'true';
  await ctx.stub.deleteState(isbn);
  console.info('===== END : Delete a Book ====='); }

```

Figure 6-8 A code snippet of the delete a book function in the smart contract

Query history of a key: This is function specifically implemented to retrieve the information on the state changes. The function accepts the ISBN as the parameter and retrieves all the state

changes that have happened to a specific key. This function returns an array of data changes occurred for the key – see figure 6- 9.

```
async getHistoryForKey(ctx, isbn) {
  console.info('===== START : QUERY ALL HISTORY FOR KEY =====');
  let iterator = await ctx.stub.getHistoryForKey(isbn);
  const allResults = [];
  while (true) {
    const res = await iterator.next();
    if (res.value && res.value.value.toString()) {
      console.log(res.value.value.toString('utf8'));

      const Key = res.value.key;
      let Record;
      try {
        Record = JSON.parse(res.value.value.toString('utf8'));
      } catch (err) {
        console.log(err);
        Record = res.value.value.toString('utf8');
      }
      allResults.push({ Key, Record });
    }
    if (res.done) {
      console.log('end of data');
      await iterator.close();
      console.info(allResults);
      return JSON.stringify(allResults);
    }
  }
  console.info('===== END : QUERY ALL HISTORY FOR KEY =====');
}
```

Figure 6-9 A code snippet of the query all history function in the smart contract

Next the library management system’s APIs are implemented.

6.2.2. API

In the current implementation, the frontend and backend layers have been segregated. Middleware, or API, is the system that connects frontend with the blockchain and IPFS networks. The API layer leverages Fabric SDK to interact and perform various operations with the backend. APIs are implemented in the RESTful fashion and JSON data structure for both request and response. Next, API functions are presented:

Register: The API layer extends the express module in NodeJS to expose http services. Register endpoint accepts the request interface and provide the response interface as an output. It accepts {name, type, passwords, email} and responds with the registration status

{status, message}. The password is stored in an encrypted function and utilises bcrypt from the crypto module of NodeJs – see figure 6-10.

```
app.post('/register', async function (req, res) {
  let response;
  if (req && req.body) {
    let data = req.body;
    try {
      const salt = bcrypt.genSaltSync(10);
      data.password = bcrypt.hashSync(data.password, salt);
      const result = await usersCollection.insertOne(data);
      console.log(`A user details was inserted with the _id:
      ${result.insertedId}`);
      response = {
        status: true,
        message: 'Registration Successful'
      };
    } catch (err) {
      response = {
        status: false,
        message: 'Failed to register the user'
      };
    }
  } else {
    response = {
      status: false,
      message: 'Failed to register the user'
    };
  }
  res.send(response);
});
```

Figure 6-10 A code snippet of the register function in API

Login: The primary objective of this endpoint is to authenticate the user and retrieve the relevant information from the database back to the application layer. The retrieved data includes user details, user type, etc. The endpoint accepts {email, password} as input and returns {status, message, data: userObject}. The password is compared with the encrypted value using the crypt method and authenticates the user – see figure 6-11.

```
app.post('/login', async function (req, res) {
  const data = req.body;
  const query = {
    email: data.email
  };

  let response;

  try {
    let userAuth = await usersCollection.findOne(query);
    const passValidation = bcrypt.compareSync(data.password,
    userAuth.password);
    delete userAuth.password;

    if (passValidation) {
```

```

        response = {
            status: true,
            message: 'User is authenticated',
            data: userAuth
        };
    } else {
        throw false;
    }
} catch (err) {
    response = {
        status: false,
        message: 'Unable to authenticate the user',
    };
}
res.send(response);
});

```

Figure 6-11 A code snippet of the login function in API

Get student: This endpoint is intended to retrieve the registered student information back to the frontend application. This data primarily used to list the student information in the application and allow librarians to issue /assign books to the students. The API does not require any parameter in the request and responds with an array of student information. The response object would include {status, message, data: data}, and data would be an array of student record [{_id, name}]- see figure 6-12.

```

app.post('/getStudents', async function (req, res) {
    const data = req.body;
    const query = {
        // hash: data.hash
        type: 'student'
    };
    let response;
    try {
        let studentData = await usersCollection.find(query).project({ 'name': 1 }).toArray();
        response = {
            status: true,
            message: 'Retrieved the data',
            data: studentData
        };
        console.log(response);
    } catch (err) {
        response = {
            status: false,
            message: 'Unable to retrieve the data',
        };
    }
    res.send(response);
});

```

Figure 6-12 A code snippet of the get student function in API

Get books: The endpoint is developed to support retrieval of listed book information from ledger and IPFS data store. The endpoint does not require input parameters, and it would produce a response with an array of books data. The response object would be {status, message, data} and the data would be an array of book information such as [{_id, bookname,

numberofcopies, yearofpublishing, author, publisher, timestamp, cidObject}]
– see figure 6-13.

```
app.post('/getBooks', async function (req, res) {
  const data = req.body;
  console.log(data);
  const query = {
    // hash: data.hash
  };
  let response;
  let bookData = [];

  try {
    const library1Client = dlt.initDlt('library1', 'user1');
    library1Client.then(async (client) => {
      console.log("Library 1 client initialized")
      await client.evaluateTransaction('queryAllBooks',
        '{"selector":{"docType":"book"}}').then((result) => {
// console.log(`Transaction has been evaluated, result is:${result.toString()}`);

        result = JSON.parse(result.toString());
        console.log(result);
        result.forEach(element => {
          console.log(element);
          console.log(element.Record);
          element = element.Record;
          let additionalData = element.additionalData;
          console.log(additionalData);
          let singleBook = {
            _id: additionalData._id,
            bookname: additionalData.bookname,
            isbn: element.isbn,
            numberofcopies: element.copies,
            yearofpublishing: additionalData.yearofpublishing,
            author: additionalData.author,
            publisher: additionalData.publisher,
            timestamp: additionalData.timestamp,
            cid: {
              path: element.DataHash
            }
          }
          bookData.push(singleBook);
        });
      });
    });
  }
  ...
});
```

Figure 6-13 A code snippet of the get book function in API

Get report: This API is limited to librarians, and it is used to fetch the report of borrowing status for the book. The API accepts data hash/ IPFS hash as the parameter and returns the report data. The response object is {status, message, data}, and the data would be an array of borrowed books and the borrowing timestamps. Data object object contains the following information: [{username, timestamp}]– see figure 6-14.

```
app.post('/getReport', async function (req, res) {
  const data = req.body;
  console.log(data);
  const query = {
    // hash: data.hash
    datahash: data.datahash
  };
});
```

```

let response;
try {
  let reportData = await transactionCollection.find(query).project({
    'userName': 1, 'timestamp': 1 }).toArray();
  response = {
    status: true,
    message: 'Retrieved the data',
    data: reportData
  };
  console.log(response);
} catch (err) {
  response = {
    status: false,
    message: 'Unable to retrieve the data',
  };
}
res.send(response);
});

```

Figure 6-14 A code snippet of the get report function in API

Add book: This endpoint is limited to librarian user roles and is primarily responsible for creating the book information in the blockchain and storing metadata in the IPFS. The API requires the parameters of {bookname, isbn, numberofcopies, yearofpublishing, author, publisher} in the request object and provides a response object that contains the status and IPFS hash. The response object would be in the form of {status, message, cid}. After receiving the request, it constructs a JSON object that has meta fields, including ISBN, title, year, author, and publisher. Then, the metadata is pushed to the IPFS using the function `ipfs.add`. In addition, the original request information and the IPFS hash are then pushed to the ledger with the key as the ISBN and the value as the data object. It utilises the `submitTransaction` method exported by the Fabric SDK— see figure 6- 15.

```

app.post('/addbook', async function (req, res) {
  console.log(req.body);
  const request = req.body;
  let data;
  let response;
  let result;
  let ipfsData = {
    "content": {
      ISBN: request.isbn,
      Title: request.bookname,
      Year: request.yearofpublishing,
      Author: request.author,
      Publisher: request.publisher
    }
  }
}

try {

  const cid = await ipfs.add(JSON.stringify(ipfsData));

  data = request;
  data.timestamp = Date.now();
  data.cid = cid;
  console.log(data.cid)
}

```

```

    result = await bookCollection.insertOne(data);
    console.log(`A book details was inserted with the _id:
    ${result.insertedId}`);
// DLT Operations
const library1Client = dlt.initDlt('library1', 'user1');
library1Client.then(async (client) => {
    console.log("Library 1 client initialised")
    await client.submitTransaction('createBook', data.isbn,
    data.numberofcopies, data.cid.path, JSON.stringify(data));
});
...

```

Figure 6-15 A code snippet of the add book function in API

Borrow book/ issue book/ return book: The borrow book and return book features are primarily intended for student roles, but to extend the functionalities, the issue book feature is made available for librarian users as well. In this endpoint, issue book is available at librarian dashboards, and using the functionality, a librarian can issue any books to the registered student. The borrow book button would be available for students in the student dashboard, and they can click this to have the book issued and assigned to their name. Once the book is borrowed, there will be an option to return the book by clicking on the return book button. The librarian will have complete visibility over the availability of books and information on who borrowed the book at any point in time. Issuance, borrow and return would be considered as transaction entries and recorded in the ledger as states. In addition, internally, it updates the availability of books whenever a borrow, issuance or return operation is triggered. The API expects parameters such as {bookname, isbn, datahash, recipientUser, operation} and generates a response object of {status, message}. Here, the operation value can be issue or return— see figure 6- 16.

```

app.post('/issuebook', async function (req, res) {
    const request = req.body;
    let data;
    let response;
    let result;

    try {
        data = request;
        data.timestamp = Date.now();
        let userMapper = await usersCollection.find({ _id:
        ObjectId(data.recipientUser) }).project({ 'name': 1 }).toArray();
        data.userName = userMapper[0].name;

        const library1Client = dlt.initDlt('library1', 'user1');

        if (data.operation == 'return') {
            // Removing from the transactions
            result = await transactionCollection.deleteOne({ "datahash":
            data.datahash, recipientUser: data.recipientUser });

            // DLT Request
            library1Client.then(async (client) => {
                await client.submitTransaction('returnBook', data.isbn,
                data.recipientUser, data.timestamp, 'Returned');
            });
        }
    }
});

```



```

        console.log("DLT Transaction Submitted");
    });
} else {
    result = await transactionCollection.insertOne(data);

    // DLT Request
    library1Client.then(async (client) => {
        await client.submitTransaction('issueBook', data.isbn,
            data.recipientUser, data.timestamp);
        console.log("DLT Transaction Submitted");
    });
}
...

```

Figure 6-16 A code snippet of the issue a book function in API

Delete book: The delete book API is limited to librarian user roles and is used to delete the book information from the ledger. The API accepts `{_id, isbn}` as parameters and deletes all the information related to the book from the ledger. The response would be a simple object with `{status, message}` – see figure 6-17.

```

app.post('/delete', async function (req, res) {
    const request = req.body;
    let response;
    let result;
    console.log(request);

    try {

        if (request.type == 'book') {
            result = await bookCollection.deleteOne({ "_id":
                ObjectId(request.id) });
            await transactionCollection.deleteMany({ "isbn": request.isbn });
            await historyCollection.deleteMany({ "isbn": request.isbn });

            console.log(`Successfully deleted the book with the _id:
                ${request.id}`);

            const library1Client = dlt.initDlt('library1', 'user1');
            library1Client.then(async (client) => {

                await client.submitTransaction('deleteBook', request.isbn);
                console.log("DLT Transaction Submitted");
            });
        }
    }
}
...

```

Figure 6-17 A code snippet of the delete a book function in API

Enrol admin: Hyperledger fabric primarily uses a PKI model to authenticate client applications and their communications. To perform any operations, such as invoke or query, in the Hyperledger fabric network, an admin identity needs to be registered. This function involves a connection profile that contains the public certificates and gRPC endpoints. Once the enrolment is successful, it will create a wallet file for the enrolled admin, which can be consumed by the application to interact with the network with admin privileges. Admin users can enrol more admins and create users with different roles in the network – see figure 6- 18.

```

async function enrollAdmin(org) {
  let adminUser = 'admin';
  try {
    // load the network configuration
    Const ccpPath = path.resolve ( _dirname, './' , 'ccp-generate',
    'connection- ${org}.json');
    Const ccp = JSON.parse(fs.readFileSync (ccpPath,'utf8'));
    ...}
  ...
}

```

Figure 6-18 A code snippet of the enrol admin function in API

Register user: Once the admin credentials are created in the network, the wallet file can be consumed by the application to enrol user credentials to the network. Any operation on the network that includes invoke or query requires a user identity to be passed. Enrolment of a user ID requires the creation of a wallet file against the user identity – see figure 6-19.

```

async function registerUser(org, user) {
  let adminUser = 'admin';
  try {
    // load the network configuration
    Const ccpPath = path.resolve ( _dirname, './' , 'ccp-generate',
    'connection- ${org}.json');
    Const ccp = JSON.parse(fs.readFileSync (ccpPath,'utf8'));
    ...}
  ...
}

```

Figure 6-19 A code snippet of the register a user function in API

Initialise gateway: Fabric blockchain utilises the gateway model to discover the network topology. Before accessing the chaincode in the network, the gateway needs to be initialised. This function helps the application to initialise the gateway and allows the application to discover the network topology. The `initDlt` function initialises an instance of the gateway, which requires a user identity to be passed to for the connection. Once the network instance is initialised, by passing the channel name and chaincode name, the contract instance can be created. Application would be able to use the contract instance to invoke or query on the blockchain network – see figure 6-20.

```

async function initDlt(org, user) {
  try {
    // load the network configuration
    Const ccpPath = path.resolve ( 'ccp-generate', 'connection-${org}.json');
    let ccp = JSON.parse(fs.readFileSync (ccpPath,'utf8'));
    // Create a new file system based wallet for managing identities.
    Const walletPath = path.join (process.cwd(), 'wallet/${org}');
    Const wallet = await Wallets.newFileSystemWallet (walletPath);
    Console.log('Wallet path:${walletPath}');
    ...}
  ...
}

```

Figure 6-20 A code snippet of the initialise gateway function in API

In the next section, frontend implementation for the case scenario is provided.

6.2.3. Frontend

The web application of the library management system case scenario consists of features to register the identity of a librarian or student, list books, query books, and enable the borrowing and returning of books. The frontend internally leverages the API layer, IPFS storage and Hyperledger Fabric network to record transactions, maintain book information and store relevant metadata. The implementation of each feature is as follows:

Login: This section is for capturing user inputs such as email address and password via the web interface. It utilises the Angular form control to have the input from the user and process it in the backend. When the user click on the submit button, it calls the `doLogin()` function to process the request. Internally it makes the API call the API service to authenticate the user— see figure 6- 21.

```
...
ngOnInit(): void {
  this.titleService.setTitle('Login - Library Portal');
  if (sessionStorage.getItem('currentUser')) {
    this.router.navigate(['/user/dashboard']);
  }

  this.loginFprm = this.formBuilder.group ({
    email: [ '', [Validators.required]],
    password: [ '', [Validators.required]], });
} ...
```

Figure 6-21 A code snippet of the login function in frontend

Register: The register page is indented to onboard new users into the system, whether librarians or students. It captures basic user information such as email, name, password, and type of user to proceed with the registration process. User type can be either librarian or student. When a user clicks on the submit button, it trigger the `doRegister()` button. Then, internally, it calls the register end point from the API service to onboard the user to the system – see figure 6-22.

```
...
ngOnInit(): void {
  this.titleService.setTitle('Login - Library Portal');
  if (sessionStorage.getItem('currentUser')) {
    this.router.navigate(['/user/dashboard']);
  }

  this.loginFprm = this.formBuilder.group ({
    name: [ '', [Validators.required, Validators.min (1)]],
```

```

type: [ 'na', [Validators.required]],
email: [ '', [Validators.required, Validators.email]],
password: [ '', [Validators.required], Validators.min (1)],});
} ...

```

Figure 6-22 A code snippet of the register function in frontend

Add book: In the current implementation, only librarians can add books into the platform. They can trigger the add book operation to get started with the book listing, and as part of the data collection, they have to provide necessary information about the book, such as name, ISBN, year of publication, author, publisher and number of copies. The issue action triggers an API call to the backend API service, which records the information in the blockchain as well as IPFS. The information is stored in the IPFS and then generates the hash. The hash and ISBN are stored in the blockchain ledger – see figure 6-23.

```

< input type= "text" fromControlName="bookname" class="form-control" id= "name"
  Placeholder= "Title" >
  </div>
  <div class = "mb-3">
    <label for= "isbn" class= "form-label" > ISBN #</label>
    <input type = "text" class "form-control" id= "isbn" placholder= "ISBN
      number" fromControlName= "isbn"
    </div>
  ...

```

Figure 6-23 A code snippet of the add book function in frontend

Issue book: As part of the implementation, a user with a librarian role can issue/assign books to the students registered in the platform. The operation internally calls the IssueBook API to perform the assignment operation in the background – see figure 6-24.

```

public doIssueBook () {
  (document.querySelector( '.btn-submit' ) as HTMLInputElement).
  setAttribute ( 'disabled', '' );

  let data: any = this.issueBookForm.value;
  data.bookname = this.singleBookName;
  data.isbn = this.singleIsbn;
  data.datahash = this.single.Datahash;
  data.operation = this.operation;
  ...

```

Figure 6-24 A code snippet of the issue a book function in frontend

Return book: Book returning is limited to student user roles. If any book is assigned to a student or they have borrowed the book in the past, then the option for return would be enabled. Once they click on return book, internally it calls the issueBook API with a type of return. On the API side, the book information would be sent back to the availability pool of books, and the available number of copies would increase by one – see figure 6-25.

```

public returnBookModal ( bookName: any, isbn: any, datahash: any) {
  this.singleBookName = bookName;
  this.singleIsbn = isbn;
  this.single.Datahash= datahash;
  this.operation = "return" ;
}

```

Figure 6-25 A code snippet of the return a book function in frontend

Delete book: Any book information recorded in the platform can be deleted by the librarian user role. The user can click on the delete button, and internally, it calls the delete API and removes all the information associated with the book from the ledger – see figure 6-26.

```

public deleteBook ( id: any, isbn: any) {
  let data = { id, type: "book" , isbn: isbn}
  const url: any = "delete ";
  this. apiService.doPostRequest(url, data). Subscribe (
  (response: any) => {
    if (response.status) {
      this.toastr.success (response.message);
      this.getBooks();
    } else {
      This.toastr.erroe (response.message);
    }
  }
  ...

```

Figure 6-26 A code snippet of the delete a book function in frontend

Next, the system dashboard will be presented.

6.2.4. Dashboard

This section presents various snapshots of user clients who interacted with system. The user can initialise a request to the API, which in turn submits the transaction to the blockchain network and IPFS storage. The network invokes the smart contract to the corresponding function to execute the transaction. When it is performed, the function returns the response to the user. Figure 6-27 shows the web dashboard utilised to register the user based on his/her role.

Full name

User type

Select a user role
 Librarian
 Student

Email address

Password

[Register](#) [Login](#)

Figure 6-27 Login page

The user can be a librarian or a student; therefore, two web dashboards are implemented, respectively – see figure 6-28.

Library Chain [Home](#) [Add new book](#) [Logout](#)

Search _____

No. ISBN	Name	Copies	Author	Publisher	Year of Publishing	Datashash	Action
1	123456Book1	10	Author1	Publisher1	2021	QmdE5i6EsEmxVV1E1ShZxJabuaPTy6WqqmDSmaYu3Gv7kV	Issue Report IPFS Delete

Items per page: 10 1 - 1 of 1 |< < > >|

© 2021 Library Chain (a)

Library Chain [Home](#) [Logout](#)

Search _____

No. ISBN	Name	Copies	Author	Publisher	Year of Publishing	Datashash	Action
1	123456 Book1	10	Author1	Publisher1	2021	QmdE5i6EsEmxVV1E1ShZxJabuaPTy6WqqmDSmaYu3Gv7kV	Borrow IPFS

Items per page: 10 1 - 1 of 1 |< < > >|

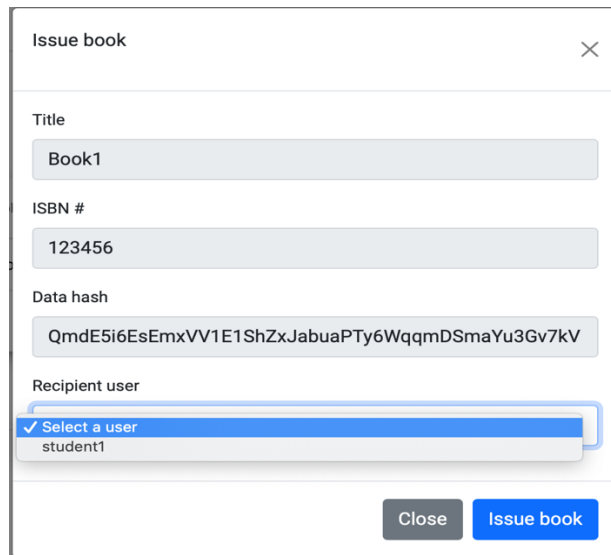
© 2021 Library Chain (b)

Figure 6-28 Snapshot of user dashboard: (a) Librarian dashboard (b) Student dashboard

The dashboard page contains information on the books listed in the system along with the availability or remaining copies. To enable search, sorting and filter features, the Material

Table Library from Angular is being utilised. It accepts a data source as a JSON input and presents data in a tabular structure where it can be sortable, filterable and searchable. The table primarily contains the ISBN, name, number of copies, author, publisher, year of publication and data hash. In addition, the librarian can perform certain operations such as adding a new book, deleting a book, checking IPFS metadata, checking the report of book borrowings and assigning a book directly to a registered user. Students can perform limited operations, such as borrow, return a book and check the IPFS metadata. In addition, the dashboard is powered with three main functions: `getBooks()`, which returns the listed books in the ledger; `getStudents()`, which returns the registered students; and `getBorrowStatus()`, which returns the borrowing status of students for each book.

The snapshot of add and assign a book dashboard, where the librarian can add the book details to the network and issue it to a registered user, can be seen in figure 6-29. The book metadata will be stored off-chain, and its ISBN will be stored on-chain along with the number of book copies and data hash. The dashboard then presents the notification generated from the blockchain to notify the user that the book has been added. It can then be assigned to a registered user.



The image shows a web form titled "Issue book" with a close button (X) in the top right corner. The form contains several input fields: "Title" with the value "Book1", "ISBN #" with the value "123456", and "Data hash" with the value "QmdE5i6EsEmxVV1E1ShZxJabuaPTy6WqqmDSmaYu3Gv7kV". Below these is a "Recipient user" dropdown menu with a blue border and a checkmark icon, showing "Select a user" and "student1" as an option. At the bottom right of the form are two buttons: a grey "Close" button and a blue "Issue book" button.

Figure 6-29 Snapshot of the add and issue a book dashboard

Figure 6-30 shows a snapshot of the borrow a book dashboard, which is available in the student dashboard. The student can view the book details and borrow it.

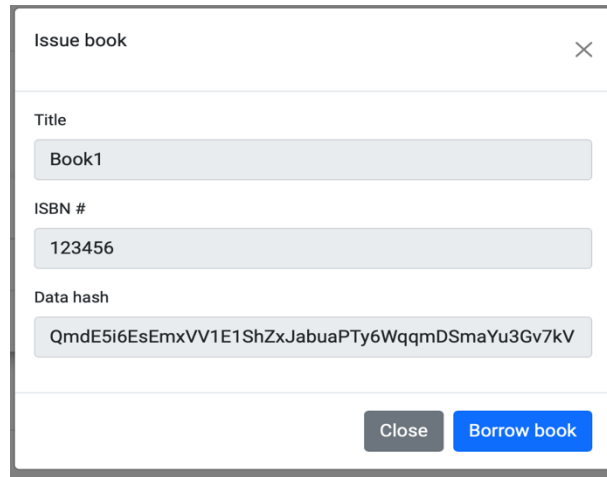


Figure 6-30 Snapshot of the borrow a book dashboard

There is an IPFS icon where both a librarian and a student can check the book metadata that is stored off-chain. Figure 6-31 presents the private IPFS network where book information is stored, and then it generates its hash data value – for example:

QmdE5i6EsEmxVV1E1ShZxJabuaPTy6WqqmDSmaYu3Gv7kV. After generating the data hash, it would be stored in the blockchain to be linked to access the information.

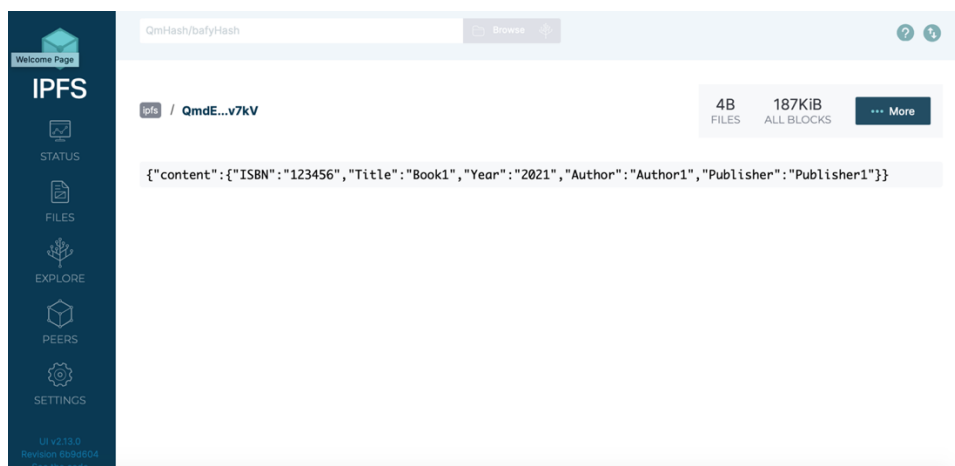


Figure 6-31 IPFS dashboard

Therefore, the above implementation of the library management system case scenario presents the use of proposed smart campus architectural framework, as can be seen in figure 6- 32.

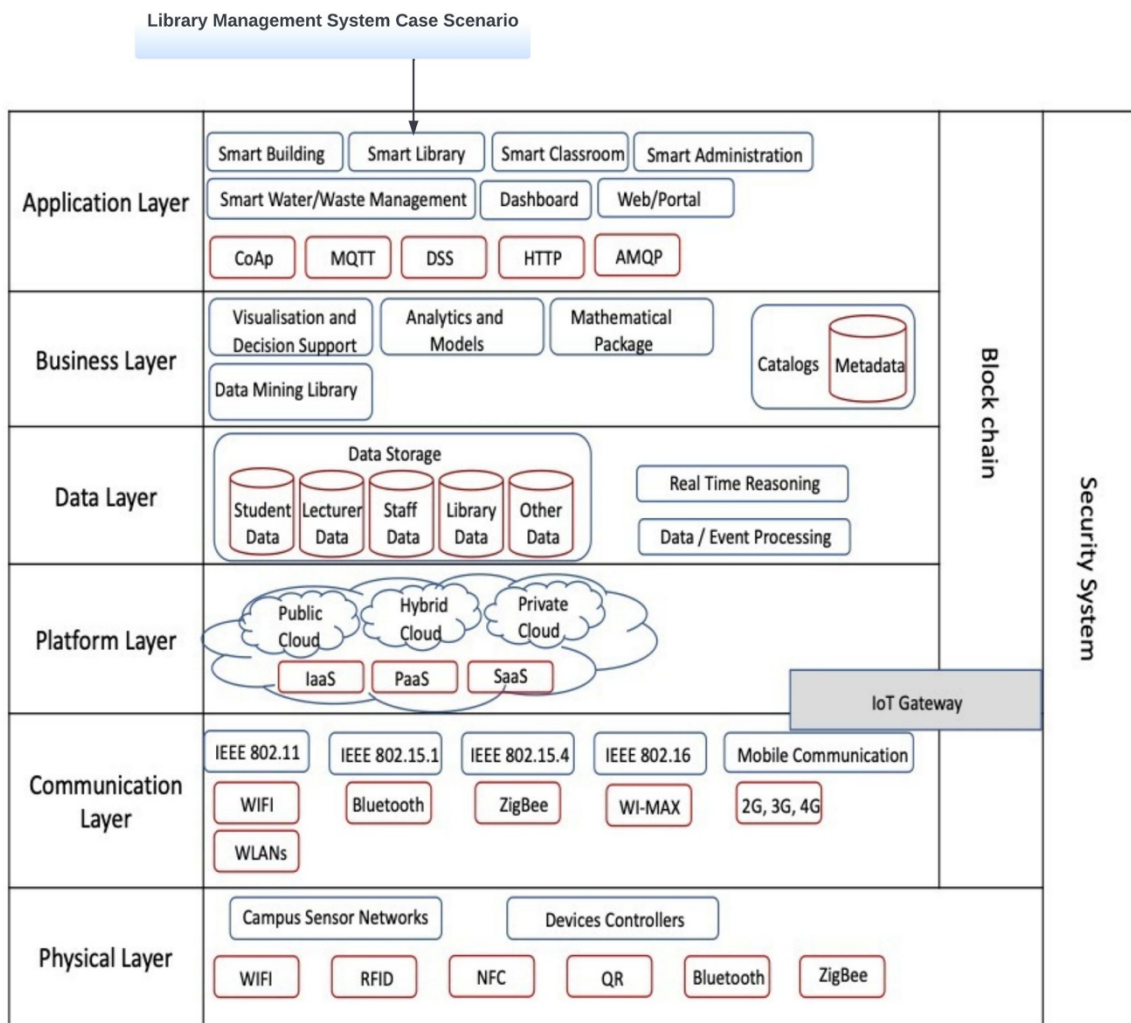


Figure 6-32 library management system case scenario based the smart campus architectural framework

The following section will develop a second proof of concept based on the proposed architectural framework.

6.3. Generating a Student Credential Report System Case Scenario

Student credentials, such as transcripts, certificates and letters of recommendation, are important documents that accompany an individual for life. Students each year gain one or more academic credentials in their portfolio. These credentials have to be issued and shared carefully. Most higher education institutions have their own department that is responsible for dealing with and managing student transcripts and certificates. Currently, educational

institutions use different approaches to ensure that credentials are tamper-proof and legitimate, including attaching security hologram labels, assigning a unique number, pasting student photos and adding a registration number [242]. Using this strategy to authenticate the credentials has become time consuming and quite complicated. In addition, similar complexities occur in verifying the credentials from any authorised entities. In applying for a job, for instance, third parties, such as a company that needs to validate a student’s certificate or transcript, may contact the educational institution for confirmation. Despite having a lengthy process, the entire system is still insecure, and fraudulent credentials are challenging to deal with. Sensitive information, such as student credentials, should be issued and shared only with a sufficient level of security and privacy.

Since the designed model is immutable as well as ensure security, privacy, and integrity to store and retrieve the data as discussed previously, it has been used for storing and sharing student records in the form of a report after verification. Figure 6-33 shows the flow diagram of the proposed solution.

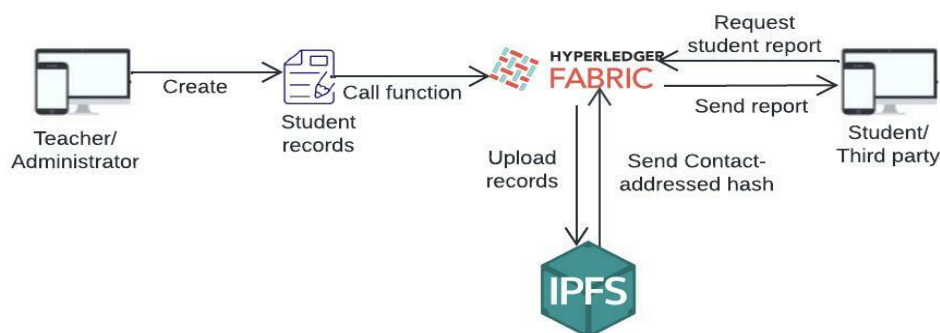


Figure 6-33 Storing and sharing student records case scenario

The diagram displays the concept of storing on-chain and off-chain as well as retrieving the data from the off-chain. The system uses the Hyperledger Fabric network that was deployed in section 6.1. The case scenario allows different stakeholders with different roles to access the system to store and share student records. There are three main actors/organisations:

- **Teacher/administrator:** This is an important actor in the use case. A university requires teachers and administrations to have permission to access, update and delete student data and records.
- **Student:** Students can access and download their record report but do not have permission to change the data.

- **Third party:** Third parties including finance, insurers and placement offices can view student reports but do not have permission to change the records. A student can share his/her credentials for claims.

In figure 6-34, a sequence diagram explains the interaction between system components and how student records are added on the network.

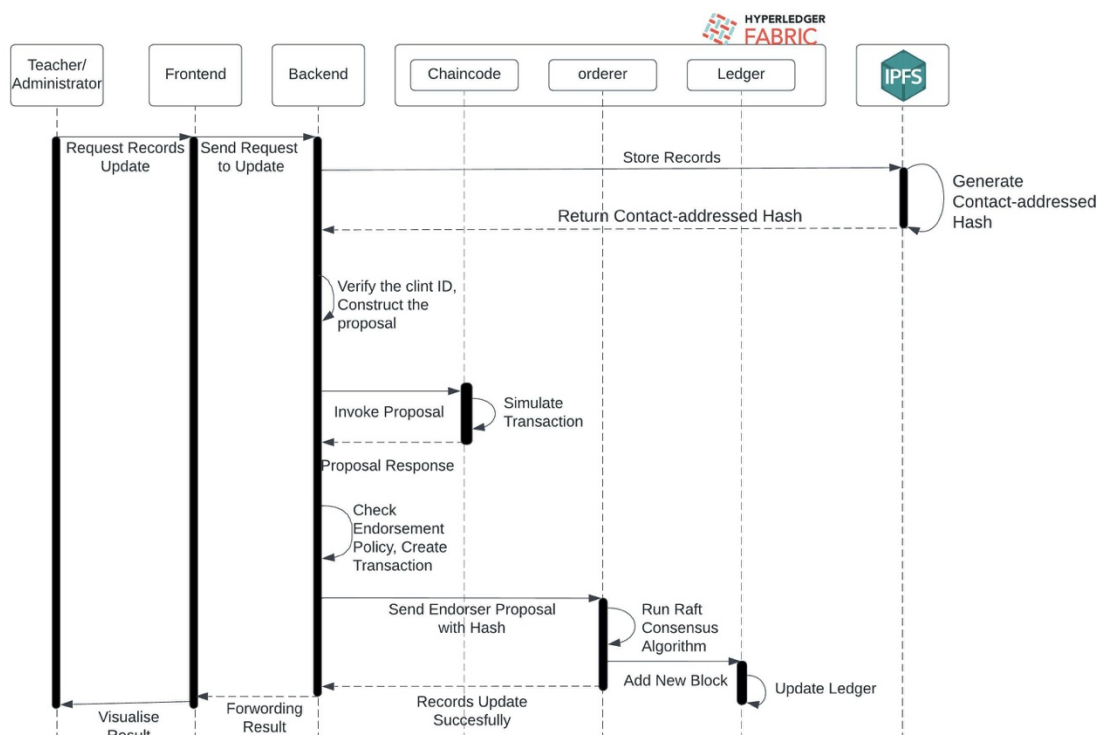


Figure 6-34 A sequence diagram for storing or updating student records

First, an authorised user from the university, such as a teacher or an administrator, requests to add a student record to the system through the application layer, with all required information such as a student ID, the student’s first name, last name, program, department, school, and the student’s records. This information is sent to the backend through the application layer. The backend segregates the data and calls IPFS to store the student’s information in a distributed fashion and return the data hash to the backend layer; the student ID will be stored later in the ledger since it will be used as a key for the stored value, to help fetch the student’s metadata from the IPFS. The backend layer creates the transactions that contains the student ID, data hash and data timestamp, and signs transactions digitally utilising the user certificate authority, which is generated by Fabric-CA. Then the transaction is sent to the Fabric Hyperledger blockchain to be stored in the ledger. The endorser peer in the blockchain is responsible for executing the smart contract, simulating the transaction’s outcome (which consists of the world

state and the read write set), and returning the outcome to the backend after appending its cryptographic signature. The backend collects all the endorser peer’s transaction responses and verifies the validity of the endorser’s signatures. After that, the backend creates the proposal and sends it to the orderers from the ordering service to run the Raft consensus algorithm, verify the transaction, and create a block to be broadcast to all nodes in the network. The peers in turn check the validity of the transaction block by verifying the endorsers’ policies and examining the read set to see whether they changed or not. After meeting all the validation requirements, the transaction block is then marked either valid or invalid, allowing the peers to update their ledger and append the transaction block in the chain. The event is sent to notify the backend that the block has been appended successfully. Finally, the application layer forwards the notification to the client and visualises the results.

In figure 6-35, a sequence diagram displays how a student retrieves and reads his or her records from IPFS, which are presented as a report.

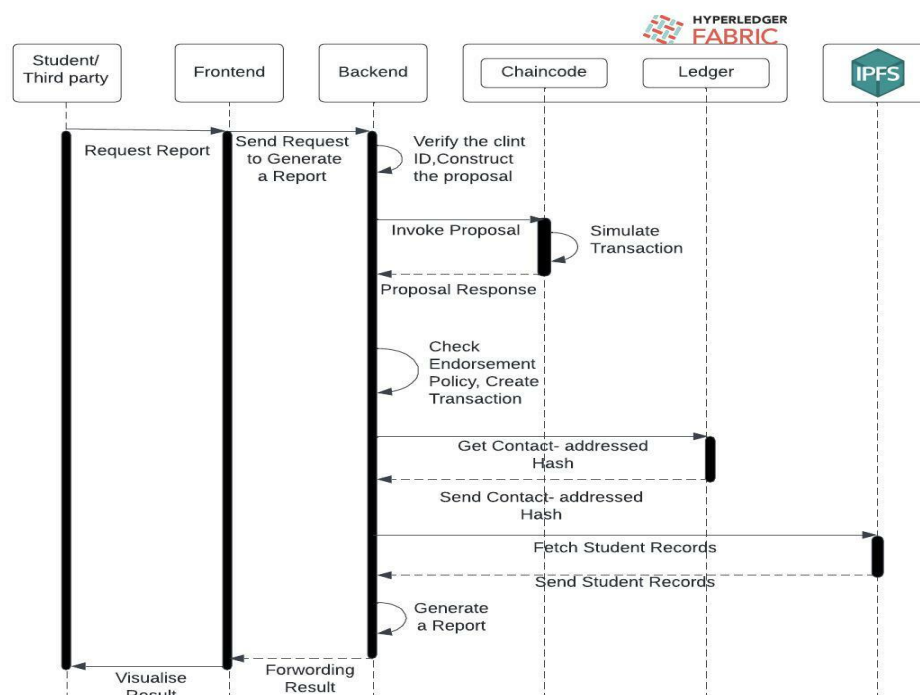


Figure 6-35 A sequence diagram for generating a student credential report

The Hyperledger Fabric blockchain discussed earlier provides several mechanisms to ensure the security and privacy of its network, such as controlling the access of each organisation or stakeholder through Fabric CA and putting chaincode restrictions on access and operation. Therefore, only a student or an authorised third party can request a student report. The

application layer will call the backend and pass the student ID to the backend in order to access its content-addressed hash that is stored in the ledger. The backend, in turn, will create a transaction proposal, which contains the student ID, and which signs it digitally, utilising the user-certificate authority generated by Fabric-CA and calling the blockchain endorsers through Fabric SDK to execute the query function in the chaincode. The endorser peer simulates the transaction's outcome, which consists of the world state and the read set, returning the outcome to the backend after appending its cryptographic signature. The backend collects all the endorser peer's transaction responses and verifies the validity of the endorser's signatures. In terms of reading a value, the value is the data hash that was generated by IPFS from the ledger; there is no need to call the ordering service because generating a block is not required. After the backend layer has the content-addressed hash, it will be sent to IPFS to fetch the student metadata. The system then presents the student information as a report that includes the status of the credentials, the data hash, the QR code (which is an encoded version of the original data), and the signatures that are generated by creating SHA256 of the original record.

The next sections present the implementation in more detail, including the scenario's chaincode API, frontend and dashboard.

6.3.1. Smart contract (Chaincode)

The case scenario is implemented with a chaincode that provides a set of functions to allow the different organisations to interact with each other. Once the chaincode is installed, all of the existing contracts will be available on its configuration channel. Each user can invoke and execute the smart contract based on his/her role. These functions are listed as follows:

Create student: The chaincode function `createStudent` is the method to insert student information into the ledger. It accepts parameters such as `ctx`, `studentId`, `signature`, `datahash` and `additionalData`. Here, `ctx` is the context of contract instance, and `additionalData` is the JSON metadata for recreating the IPFS data. `ctx.stub.putState` is the function used to create the state in the ledger. It accepts a key and a value as parameters, where the key is `studentId` and the value is a JSON string of the student data, which includes the records – see figure 6- 36.

```
async createStudent(ctx, studentId, signature, datahash, additionalData) {
  console.info('===== START : Create Student =====');
  additionalData = JSON.parse(additionalData);
  const student = { StudentId: studentId, docType: 'student', signature:
  signature, DataHash : datahash, additionalData:additionalData
```

```

};
await ctx.stub.puttState(srudentId, Buffer.from (JSON.stringify(student)));
console.info('===== END : Create Student =====');
}

```

Figure 6-36 A code snippet of the create a student function in the smart contract

Query student: This function retrieves student data from the ledger by passing the studentId. It uses the getState method to retrieve the state from the ledger. This function returns a student data object which contains {studentid, firstname, lastname, program, department, school, attendance, transcripts, cid}, as in figure 6-37.

```

async queryStudent(ctx, studentId) {
  console.info('===== START : QUERY A STUDENT =====');
  const studentAsBytes = await await ctx.stub.getState (studentId); // get
  the student from chaincode state
  if (!studentAsBytes || studentAsBytes.length ===0) {
    throw new Error(`${studentId} does not exist`);
  }
  console.log(studentAsBytes.toString());
  console.info('===== END : QUERY Student END =====');
  return studentAsBytes.toString();
}

```

Figure 6-37 A code snippet of the query a student function in the smart contract

Query All student: This function is responsible for querying the student information from the ledger. The function accepts parameters such as ctx and query, where query is a JSON object with selector or filtering parameters. This function returns the array of student data as a response: [{studentid, firstname, lastname, program, department, school, attendance, transcripts, cid }]- see figure 6-38.

```

async queryAllStudent(ctx, query) {
  console.info('===== START : QUERY ALL STUDENTS TRANSACTIONS =====');
  let iterator = await ctx.stub.getQueryResult(query);
  const allResults = [];
  while (true) {
    const res = await iterator.next();
    if (res.value && res.value.value.toString()) {
      console.log(res.value.value.toString('utf8'));
      const Key = res.value.key;
      let Record;
      try {
        Record = JSON.parse(res.value.value.toString('utf8'));
      } catch (err) {
        console.log(err);
        Record = res.value.value.toString('utf8');
      }
      allResults.push({ Key, Record });
    }
  }
  if (res.done) {
    console.log('end of data');
  }
}

```

```

        await iterator.close();
        console.info(allResults);
        return JSON.stringify(allResults);
    }
}
console.info('===== END : QUERY ALL STUDENTS TRANSACTIONS=====');
}

```

Figure 6-38 A code snippet of the query all students function in the smart contract

Delete student: This method deletes student records from the ledger. The function accepts `studentId` as an input and removes the entered student information from the blockchain. However, the student data is not actually deleted – it remains intact, as the transaction is marked as deleted so as not to appear in the system. In the ledger’s history, it still exists – see figure 6-39.

```

async deleteStudent(ctx, studentId) {
    console.info('===== START : Delete Student =====');
    const studentAsBytes = await ctx.stub.getState(studentId); // get the
    student from chaincode state
    if (!studentAsBytes || studentAsBytes.length === 0) {
        throw new Error(`${isbn} does not exist`); }
    await ctx.stub.deleteState(studentId);
    console.info('===== END : Delete Student ====='); }

```

Figure 6-39 A code snippet of the delete a student function in the smart contract

Query history of a key: This function retrieves information on state changes. The function accepts `studentid` as the parameter and retrieves all the state updates that have happened to a specific key. This function returns an array of data changes that have occurred for the key– see figure 6-40.

```

async getHistoryForKey(ctx, studentId) {
    console.info('===== START : QUERY ALL HISTORY FOR KEY =====');
    let iterator = await ctx.stub.getHistoryForKey(studentId);
    const allResults = [];
    while (true) {
        const res = await iterator.next();
        if (res.value && res.value.value.toString()) {
            console.log(res.value.value.toString('utf8'));

            const Key = res.value.key;
            let Record;
            try {
                Record = JSON.parse(res.value.value.toString('utf8'));
            } catch (err) {
                console.log(err);
                Record = res.value.value.toString('utf8');
            }
            allResults.push({ Key, Record });
        }
    }
}

```



```

        if (res.done) {
            console.log('end of data');
            await iterator.close();
            console.info(allResults);
            return JSON.stringify(allResults);
        }
    }
    console.info('===== END : QUERY ALL HISTORY FOR KEY =====');
}
}

```

Figure 6-40 A code snippet of the query all history function in the smart contract

6.3.2. API

As per the first case scenario, the API was used as a middleware layer to connect the frontend with the blockchain and IPFS networks. It leverages Fabric SDK to interact and perform various operations with the backend. In addition, APIs are implemented in the RESTful fashion and JSON data structure for both request and response. APIs are presented in more detail as follows:

Register: For API register user, the same implementation in figure 6-9 is used.

Login: For login, see figure 6-10 where the same login API is used.

Add student: The endpoint is responsible for creating user credentials in the system. It internally requests relevant calls to the IPFS for storing the data and Hyperledger Fabric for inserting the data hash. This endpoint accepts data in the form of a JSON structure {studentid, firstname, lastname, program, department, school, attendance, transcripts} where the transcript is an array of objects in the form of {moduleTitle, mark}. The code internally aggregates the scores and then calculates the percentage. If the percentage is above 70, the outcome would be first class. Similarly, if the percentage is above 60, 50 or 40, the outcomes would be upper second class, lower second class and third class, respectively— see figure 6-41.

```

app.post('/addStudent', async function (req, res) {
    console.log(req.body);
    const request = req.body;
    let data;
    let response;
    let result;
    let ipfsData = {
        "content": {
            studentid: request.studentid,
            firstname: request.firstname,
            lastname: request.lastname,
            program: request.program,
            school: request.school,
            attendance: request.attendance,
            transcripts: request.transcripts
        }
    }
}
}

```



```

try {
    const cid = await ipfs.add(JSON.stringify(ipfsData));
    data = request;
    data.timestamp = Date.now();
    data.cid = cid;
    // Hashing it for the info
    Data.signature= crypto.createHash ('sha256').update(ipfsData).
    digest('hex');
...}
...

```

Figure 6-41 A code snippet of the add student function in the API

Get student: This endpoint is intended to retrieve the student information back to the frontend application. This data is primarily used to list the student information and create credentials to appear for the student. The API does not require any parameters in the request and responds with an array of student information. The response object is in the form of {status, message, data: data}, and the data would be an array of student records [{studentid, firstname, lastname, program, department, school, attendance, transcripts, cid}], where the transcript is an array of objects in the form of {moduleTitle, mark}– see figure 6- 42.

```

app.post('/getStudents', async function (req, res) {
    const data = req.body;
    console.log(data);
    const query = {
        // hash: data.hash
    };
    let response;
    let studentData = [];
    ...
}...

```

Figure 6-42 A code snippet of the get student function in the API

Delete record: The delete student API is limited to the admin user role and is used to delete the student information from the ledger. The API accepts {_id, studentid} as parameters and deletes all the information related to the student from the ledger. The response would be a simple object with {status, message}– see figure 6-43.

```

app.post('/delete', async function (req, res) {
    const request = req.body;
    let response;
    let result;
    console.log(request);

    try {
        result = await studentCollection.deleteOne
        ({ "_id": ObjectId(request.id)});
        Console.log ('successfully deleted the student with the _id:
        ${request.id}');
    ...

```

```
});
```

Figure 6-43 A code snippet of the delete record function in the API

Credential verification: Credential verification endpoint accepts `studentid` as the input and sends it to the ledger to retrieve the data hash. Then, it fetches the data that is stored in the IPFS using the data hash. The function then internally computes the hash of the data to ensure that the data is digitally verified. This function also generates a Quick Response (QR) code, which contains the credential information as an additional feature – see figure 6-44.

```
app.post('/verify', async function (req, res) {  
  ...  
  Function (dltResult) {  
    Let studentData= JDON.parse(dltResult.toString());  
    if (studentData.length > 0) {  
      StudentData= studentData[0];  
    }  
    if (studentData && studentData.DataHash) {  
      Let ipfsCID= studentData.DataHash;  
      Axios.get(`${IPFS_URL}:8080/ipfs/` +ipfsCID)  
        .then (resonse=> {  
          Let ipfsResponse = response.data;  
        }  
    }  
  }  
  ...  
}
```

Figure 6-44 A code snippet of the credential verification function in the API

Initialise gateway: The gateway needs to initialise to be able to access the blockchain. See figure 6-19 for more details.

The following section presents the frontend implementation for the case scenario.

6.3.3. Frontend

The frontend of the generating student report case scenario has a web application to interact with the backend. The web application internally leverages the API layer, IPFS storage and Hyperledger Fabric network to record transactions, maintain student information and store relevant metadata. It consists of features to register the identity of an administrator or student, create student records and search or query student records, as well as a certificate search page and a credential verification page. The implementation of each feature is explained in more detail below:

Login: For the login user function, the same implementation in figure 6-20 was used.

Register: To register a new user, whether an admin or a student, in the system, the `doRegister()` function has been implemented. The submit button triggers the function that in

turn calls the register end point from the API service to register the users. Basic information is needed, such as name, email, password and user role – see figure 6-45.

```
<div class= "card align-middle">
  <div class= "card-body">
    <form [formGroup]= "registerForm" (ngSubmit)= "doRegister()"
      Autocomplete= "off">
      <div class = "mb-3">
        <label for= "name" class= "form-label" > Full name </label>
        <input type = "text" formControlName= "name" class= "form-
          control" id= "name" placeholder= "Full name">
      </div> ...
```

Figure 6-45 A code snippet of the register function in frontend

Add student: This functionality is primarily for the admin of the institution. By using this functionality, one can create a student record by populating information such as student ID, name, academic institution, department, course and transcript. The transcript is powered by a dynamic form where the user can add any number of modules and achieved marks. Internally, the system calculates the outcome and the overall percentage – see figure 6- 46.

```
public doAddStudent(){
  if (this.addStudentForm.invalid) {
    this.observables.changeformValid (true);
    return;
  } else {
    (document.querySelector('.btn-submit') as
      HTMLInputElement).setAttribute ('disabled', ' ');
    const data: any = this.addStudentForm.value;
    console.log (data);
    const url: any = "addStudent";
    this.apiService.doPostRequest(url,data).subscribe((response: any) =>
  { ...
```

Figure 6-46 A code snippet of the add student function in frontend

Search credential: The search page is primarily intended for students and authorised third parties who require access to credentials. This screen accepts unique student IDs and retrieves the credentials from the backend. Internally, the system verifies the hash of the data and ensures the credentials are digitally valid – see figure 6- 47.

```
...
public doStearch(){
  const data: any = this.searchForm.value;
  if (data && data.search) {
    this.router.navigate (['/user/verify/' + data.search]);
  }...
}
```

Figure 6-47 A code snippet of the search credential function in frontend

Credential verification Page: This page is the result of the search credential page. It carries the status of the credential, data hash, QR code, which is an encoded version of the original

data, and the signatures that are generated by creating SHA256 of the original record – see figure 6-48.

```

Public fetCertificate() {
  const url: any = "verify";
  Let data = {
    studentid: this.studentId
  }
  This.apiService.doPostRequest(url,data).subscribe ((response:any) => {
    if (response.sratus) {
      this.studentData = response.data;
      this.signature = response.signature;
      this.verificationStatus = response.verificationStatus;
      this.qrcode= response.qrcode;
      ...
    }
  })
}

```

Figure 6-48 A code snippet of the search credential page function in frontend

In the next section, the dashboard of the case scenario is provided.

6.3.4. Dashboard

In this section, several snapshots of user clients who interacted with the system are presented. The user can initialise a request to the API, which in turn submits the transaction to the backend, including the blockchain network and IPFS storage. The network invokes the smart contract to the corresponding function to execute the transaction. When performed, the function returns the response to the user. In this case scenario, the user can be an administrator, student or third party such as finance, insurers and placements office. Therefore, two web dashboards were implemented, respectively – see figure 6- 49.

No.	Student ID	First Name	Last Name	Program	Department	School	Total Mark	Outcome	Datahash	Action
1	123456	Student 1		Bachelor of Computer Science	Informatics	School of Engineering and Informatics	78.33%	First Class	Qme1CgFvxCPDB1KcT32H34igc28r1YuJFzeSUKMRi8tZef	View IPFS Delete
2	123457	Student 2		Bachelor of Computer Science	Informatics	School of Engineering and Informatics	83.33%	First Class	QmXChszqaQ3BM4YrsCKxuN4BKPYPk1bbdQmsEbSsz1tVdLS	View IPFS Delete

(a)

Search

No.	Student ID	First Name	Last Name	Program	Department	School	Total Mark	Outcome	Datahash	Action
1	123457	Student	2	Bachelor of Computer Science	Informatics	School of Engineering and Informatics	83.33%	First Class	QmXChszqaQ3BM4YrsCKxuN4BKPYpK1bbdQmsEbSz1tVdLS	View IPFS

(b)

Figure 6-49 Snapshot of user dashboard: (a) Admin dashboard (b) Student or third-party dashboard

The above dashboard page contains information on the student credentials created in the system along with academic information. The dashboard uses Material Table Library from Angular to enable search, sorting and filter features. It accepts a data source as a JSON input and presents data in a tabular structure where it can be sortable, filterable and searchable. The table primarily contains the student ID, name, program, department, school, total mark, outcome and data hash. In addition, admin can perform certain operations such as adding student records, deleting students and checking IPFS metadata. Students can view the digitally verified credentials and check the IPFS metadata. The dashboard is powered with functions such as `getStudents()`, which returns the listed students in the ledger.

The snapshot in figure 7-50 presents the adding a student record dashboard, where the admin can add student details and records to the network. The student’s academic metadata will be stored off-chain, and the student’s ID will be stored on-chain along with the data hash generated in the IPFS. The dashboard then presents the notification generated from the blockchain to notify the user that the student has been added.

Student ID

First Name Last Name

Award Programme

Department School

Mode of Attendance

[Add Modules](#)

Module Title Mark %

Figure 6-50 Snapshot of the adding a student records dashboard

The search page snapshot in figure 6-51 is for students and authorised third parties who require access to credentials. This screen accepts unique student IDs and retrieves the credentials from the backend after verification.

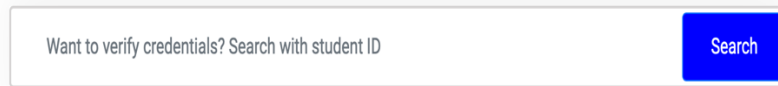


Figure 6-51 Snapshot of the search page

The IPFS button is where both the admin and the student or third-party can view student metadata stored off-chain, such as student academic information and student credential information. Figure 6-52 presents the private IPFS network where student information is stored, and it then generates its hash data value. After generating the data hash, it would store data in the ledger to be linked to access the information.



Figure 6-52 IPFS dashboard

The system can view and download the student credential report that includes the student's information, status of credentials, data hash, QR code, which is an encoded version of the original data, and the signatures that are generated by creating SHA256 of the original record – see figure 6-53.

Academic Student Report

Report Date: Tuesday, May 3, 2022

Student Details

Student ID	2345678
First Name	Student
Last Name	2

 Verified
Credential

[Download Certificate](#)

Academic Details

Award Programme	Bachelor of Computer Science
Department	Informatics
School	School of Engineering and Informatics
Mode of Attendance	Full-time



Academic Result

Module Title	Mark %
Module1	70%
Module2	60%
Module3	65%
Overall Result	66.00%
Outcome	Upper Second Class

Verification Signature:

3d6963b790605a405f0fbc50da6e5165e28917f8289b3ced8ac273ec458b0ff

Verification URL: <http://3.111.72.254:7051/user/verify/2345678>

IPFS URL:

<http://3.111.72.254:3000/#/ipfs/QmZuh5t9FGQw8MDA2zBzBUXfWgvXyeuLpdrpFYJgw7pTGd>

Figure 6-53 Snapshot of the student credential report

In case the student data has been tampered, the hash validation will be failed between the blockchain and IPFS data store. Thus, the system can view the student credential report with invalid verification – see figure 6-54.

Academic Student Report
Report Date: Tuesday, May 3, 2022

Student Details

Student ID	2345679
First Name	Student
Last Name	3

Academic Details

Award Programme	Bachelor of Computer Science
Department	Informatics
School	School of Engineering and Informatics
Mode of Attendance	Full-time

Academic Result

Module Title	Mark %
Module1	70%
Module2	60%
Module3	60%
Overall Result	63.33%
Outcome	Upper Second Class

Verification Signature:
c35db451f49f8020ae319e1e03b968b2ab5106d8b5b5334d982174c7333e2b2d |
80d439db4f6f4c18bcdcdca13e23af68dee47c894dc1b686f15d899fcbe77158

Verification URL: <http://3.111.72.254:7051/user/verify/23456>

IPFS URL:
<http://3.111.72.254:3000/~/ipfs/QmThcr423kWjuIDE48khzd2R2i19jEJFSogbx1XfzdUhoo>

❌ Tampered
Credential

Download Certificate




Figure 6-54 Snapshot of the student invalid credential report

After implementing the second case scenario of generating a student credential report, including storing student records and generating and sharing a student report, figure 6-55 shows where it can be pluggable in the proposed smart campus framework.

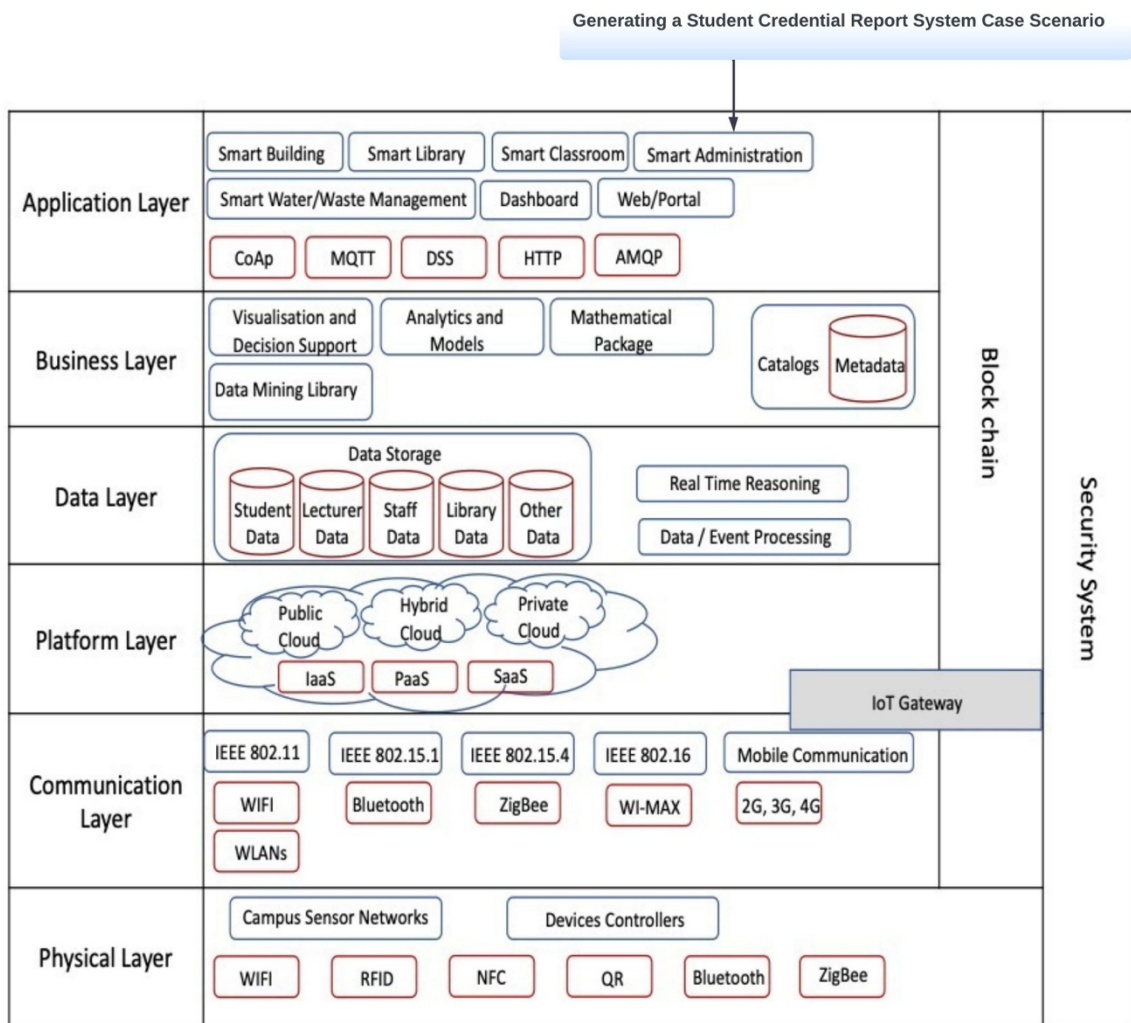


Figure 6-55 Generating a student credential report case scenario based the smart campus architectural framework

In the next section, the proposed architectural framework will be evaluated using a series of experimental tests.

6.4. Evaluation

To evaluate the proposed system, a series of experimental tests are carried out utilising different software tools to provide comprehensive results and to demonstrate the applicability of the framework. Several use case scenarios have been conducted with varying configurations to present different indicators. The indicators in this analysis for the network performance are throughput, which is defined as successful transactions completed in a second; transaction

latency, which is the time that a transaction takes between submitting and receiving a response; and resource utilisation, which is visualised through graphs. According to Pongnumkul et al. [243], throughput and latency are the main quantifications to understand the existing blockchain performance and its limitations.

For testing the blockchain network, Hyperledger Caliper [244] is used as a blockchain benchmark tool, which was developed by the Linux Foundation. This tool allows users to compute the network performance by configuring a use case script to generate the workload of a particular blockchain application and then generate reports, which are displayed as metrics. It stimulates a real-world application case scenario by sending all generating transactions in parallel to launch the required load, thereby displaying the performance reports. ‘Caliper architecture’ consists of three main layers (see Figure 6-56): the benchmark layer, which is responsible for defining test factors to evaluate the backend blockchain performance; the core layer, which plays a role in generating a performance report; and the adapter layer, which is responsible for the interaction between the Caliper framework and the Hyperledger Fabric blockchain network. With the help of Docker containers, Caliper is developed in the same virtual machine used to conduct the experiments – see Table 6-1. In addition, benchmark-config.yaml was defined that contains benchmarking tests and defining workload file that has the workload functions in JavaScript. The source code of the testing can be found in

<https://github.com/Manal979/Project/tree/master/caliper>

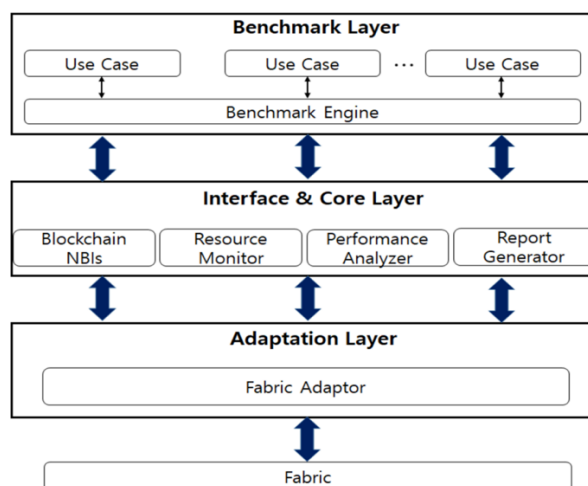


Figure 6-56 Hyperledger Caliper architecture [244]

To test the network performance, vary transaction rates are conducted in each executed round. There are six rounds, and each round has a different transaction rate, ranging from 50, 100,

150, 200, 250 and 300 transactions per second (TPS) – small to mid-size universities support around 300 TPS. The fixed number of total transactions in each round is 1,000. In each round, the average transaction latency and transaction throughput are measured to be presented in the graph plots. Different transaction mode scenarios are used for testing, including writing transaction mode, reading transaction mode and writing and reading set transaction mode.

As can be seen in table 6-2, the first testing case is for evaluating writing transactions where a ledger needs to be updated with random values by calculate transaction latency and throughput. Writing transaction latency can be presented as a mathematical formula:

$$WT_L = (T_c * N_T) - S_T$$

Where WT_L is the time that the transaction takes for utilising the smart education network, T_c presents the time of transaction confirmation, N_T is the network threshold change and S_T is the transaction submit time. Transaction throughput can be presented as:

$$WT_T = T_{CT} / T_{TS} * N_{CN}$$

Where WT_T is the number of successful writing transactions per second, T_{CT} is the committed transaction in the system and T_{TS} is the completed transactions, which is the result of subtracting the failed transactions from the total transactions at N_{CN} committed participates.

Table 6-2 Testing the writing transactions mode

Parameter	Configuration
Number of Rounds	six
Total transactions	1000 per rounds
Transaction Rates	50, 100, 150, 200, 250, 300 TPS
Transaction mode	Write

The below figure presents the average latency and throughput against the transaction rates for the writing transactions mode, where the latency is measured in seconds.

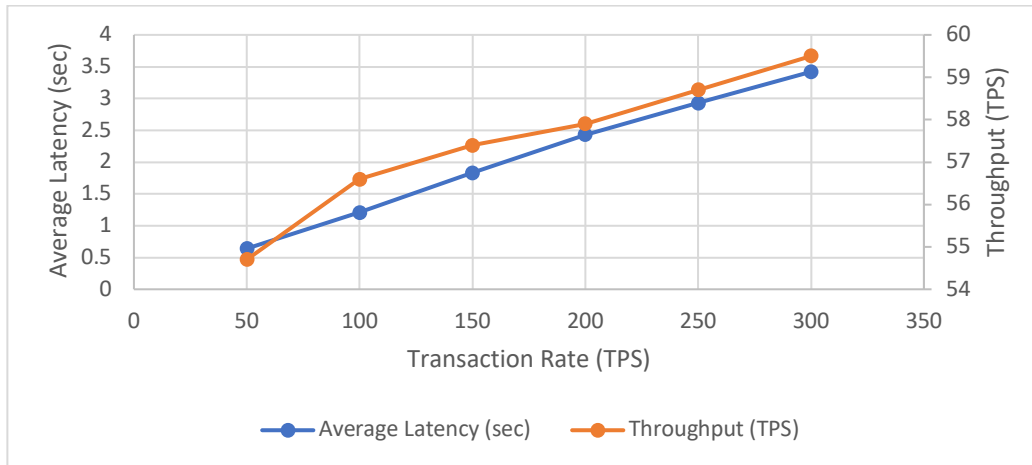


Figure 6-57 Evaluating the first scenario: writing transactions mode on the network latency and throughput

In each round, when the writing transaction rate rises, it leads to a gradual increase in the average latency time. The average throughput almost linear after 200 TPS. Generally, low latency leads to higher throughput.

Our second test scenario focuses on evaluating reading or querying transactions by calculating transaction latency and throughput. In this scenario, all clients sending their query at the same time to a single node to generate the required read workload. The latency for reading transactions can be calculated by the mathematical formula:

$$RT_L = T_R - S_T$$

Where RT_L is reading transaction latency time, S_T is submitting time and T_R is response time when received. Reading transaction throughput is the total number of query transactions achieved in a second, which can be presented as:

$$RT_T = T_o - T_s. T_o$$

measure the reading transaction throughput, RT_T , the total number of query or reading transactions, T_o , is subtracted from the total time in seconds, T_s – see table 6-3.

Table 6-3 Testing reading transactions mode

Parameter	Configuration
Number of Rounds	six
Total transactions	1000 per rounds
Transaction Rates	50, 100, 150, 200, 250, 300 TPS
Transaction mode	Read

The figure 6-58 presents the average latency and throughput against transaction rates for reading transactions mode. The latency in the figure is measured in seconds.

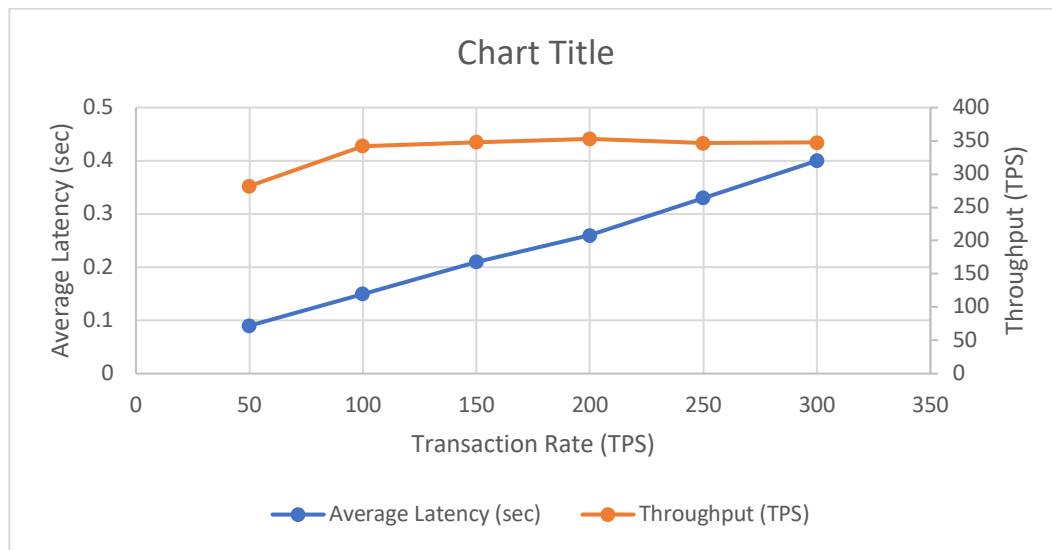


Figure 6-58 Evaluating the second scenario: reading transactions mode on the network latency and throughput

The evaluation results of the above figure indicate that the average latency increases without a major rise, while the transaction rates increase in each round. The average latency is near zero, and the system can handle 300 TPS without delays. This suggests that the system does not reach its maximum capacity and can support higher transaction rates. In addition, the increasing transaction rates had no significant effect on the average throughput. The throughput was almost flat in all rounds above 100 TPS. By submitting a varying load of reading transactions, the number of GET REST API calls increased in order to fetch data from both on-chain and off-chain, which caused a flattening throughput in this system capability.

After testing the writing transactions and reading transactions against transaction rates of the proposed system, they can be compared. The average latency in the writing transaction mode is higher than the average latency in the reading transaction mode. This is because the writing transaction needs to verify the transaction and update the ledger by executing the smart contract and running the consensus algorithm. In contrast, the reading transaction requires execution of the smart contract to gain the information stored in the blockchain and IPFS.

Our third test scenario focuses on evaluating reading and writing set transaction mode by calculating transaction latency and throughput. In this scenario, a mix of submitting read and write transactions was used. The clients send write transactions while they are in process, the read transactions are submitted to receive responses – see table 6-4.

Table 6-4 Testing writing and reading transactions mode

Parameter	Configuration
Number of Rounds	six
Total transactions	1000 per rounds
Transaction Rates	50, 100, 150, 200, 250, 300 TPS
Transaction mode	Read, Write

The figure 6-59 presents the average latency and throughput against transaction rates for writing and reading transactions mode.

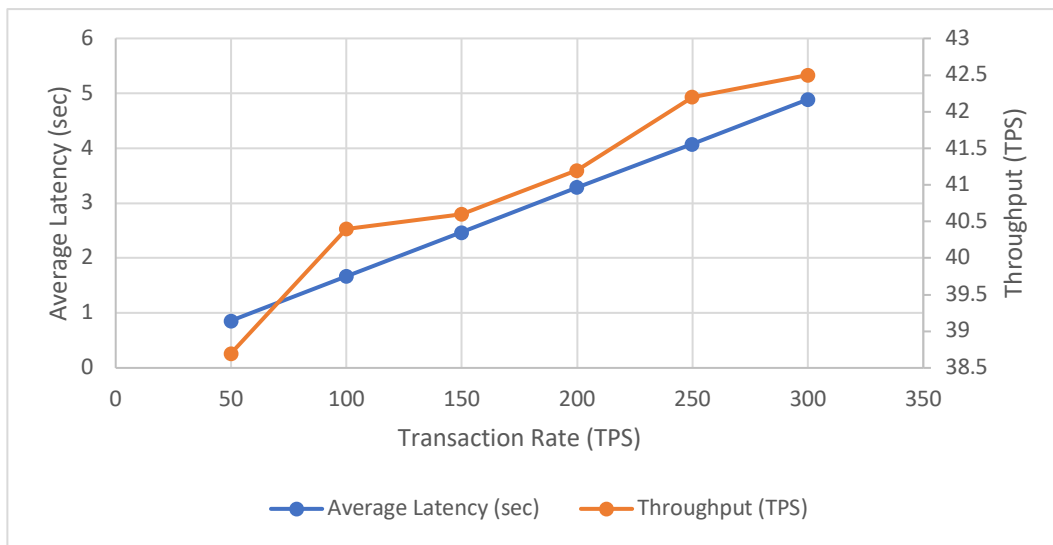


Figure 6-59 Evaluating the third scenario: writing and reading transactions mode on the network latency and throughput

Therefore, the system performance results showed that the system is lightweight since it avoids high network latency and low throughput. In addition, the system achieves the quality requirements for smart education systems that was discussed in section 4.1 as proof of concept for the architectural framework of a smart campus. The next section will present the future applicability of the framework in a broader context.

6.5. Broader Context Case Scenario: Developing a Framework for the Adoption of Blockchain in the Higher Education Certification Process in Saudi Arabia

This section uses the proposed comprehensive framework for a smart campus (see figure 3.1) in a broader context to test its future applicability. The framework uses to develop a student certification framework, with higher education in Saudi Arabia acting as a case study [108]. This work has been produced and published in collaboration with another PhD student named Mona Alshahrani.

Over recent decades, the expansion of research and development programs, alongside the growth of new educational schemes, has noticeably improved Saudi Arabia's educational system. The Saudi Ministry of Education (MoE) has numerous key aims, one of which involves establishing an integrated services system that supports educational processes by improving performance and efficiency and promoting state-of-the-art technologies. In 1999, Saudi Arabia had only ten universities: by 2017, this figure had risen to 26, with the expectation being that more institutions will be set up to mirror Saudi Arabia's population growth. According to estimates, the country's 2012 population of 29.2 million would increase to 35.9 million by 2020. In 2016, the number of university-age students in Saudi Arabia amounted to approximately 1.7 million, showing a marked increase from the 850,000 and 650,000 students recorded in 2009 and 2006, respectively. Such developments indicate that the Saudi higher education system has experienced significant changes in its capacity, international connections, graduate outcomes and research impacts. Therefore, this joint paper aims to fill a gap in the applicability of the novel framework particularly in developing smart educational environments such as certification or credentialing systems in Saudi Arabia. the study posits that the proposed framework's integration of all relevant actors, process and storage units shows its legitimacy; the framework also logically accounts for the principal obstacles in current systems, such as certificate fraud and dishonesty.

Blockchains also exploit the property of immutability for student records, which the study believes will address critical challenges in current higher education institution credentialing processes. The existing design phase does not highlight any data storage concerns in a real-world setting, which creates difficulties in analysing system scalability [108]. Under a scenario

where a blockchain acts as a database to store student records, such as ID, name, date of birth, department, courses and badges achieved, the number of data points would overwhelm the chain and spread across the network's nodes. Eventually, such a blockchain would encounter challenges maintaining and storing this information, potentially compromising system performance.

Blockchain technology eliminates the need for third-party intermediaries and improves interactions between participants (as discussed in section 3.2). To resolve the identified problems and challenges in existing certifying systems in higher education, the second co-author, Mona, proposed the structure and functionality of DASC based on the novel smart campus architectural framework. According to Figure 6-60, the DASC system comprises five principal actors: instructors, students, administrators, alumni and prospective employers.

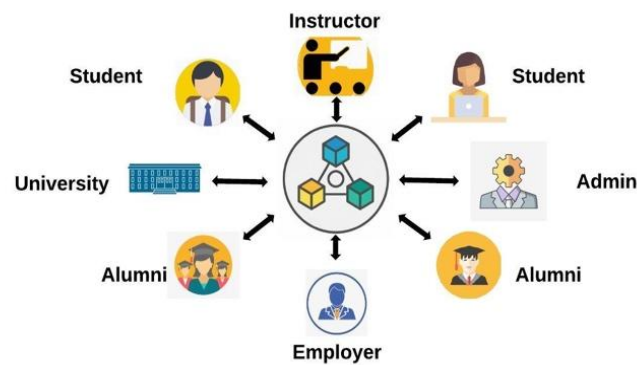


Figure 6-60 Actors in DASC [108]

DASC [108] aims to maintain a log of student data, including skills, credits, badges and course registrations. The system should have the capacity to share student data with authorised stakeholders, such as prospective employers, university staff and university administrators. The system should also have a high level of transparency that enables higher education institutions to design and implement distinctive, personalised teaching methods for each student.

Additionally, the DASC system should function as a standard information repository that collates students' information – including digital certificates, achievements and transcripts– from different higher education institutions. Such a system would allow students to maintain authentic records in a long-term e-portfolio logging their certificates, courses, grades and achievements. Given that prospective employers could use the proposed system to verify the authenticity of a candidate's qualifications and transcript, it will eliminate certificate fraud and dishonesty.

The second co-author also conduct a survey to help improving the initially proposed model. Quantitative analysis indicated that the four influential factors (security and privacy, trust, social influence and efficiency) substantially impacted the acceptance of blockchain technology by students and prospective employers. This development illustrates the model structure's validity, readiness for implementation and capacity to test user feedback.

Figure 6-61 illustrates the DASC systems's high-level conceptual infrastructure, with the blockchain represented as the left dashed box as on-chain transactions, connected to the front-end system and centralised database systems as off-chain transactions. On-chain transactions occur directly on the distributed ledger network, while external off-chain transactions occur outside the distributed ledger. DASC system means students can view their credentials with a high level of integrity in a single location and decide whether to share this view with other stakeholders. According to the conceptual infrastructure, the DASC system enables interactions between prospective employers and front-end systems, overseen by system administrators who allocate the correct permissions.

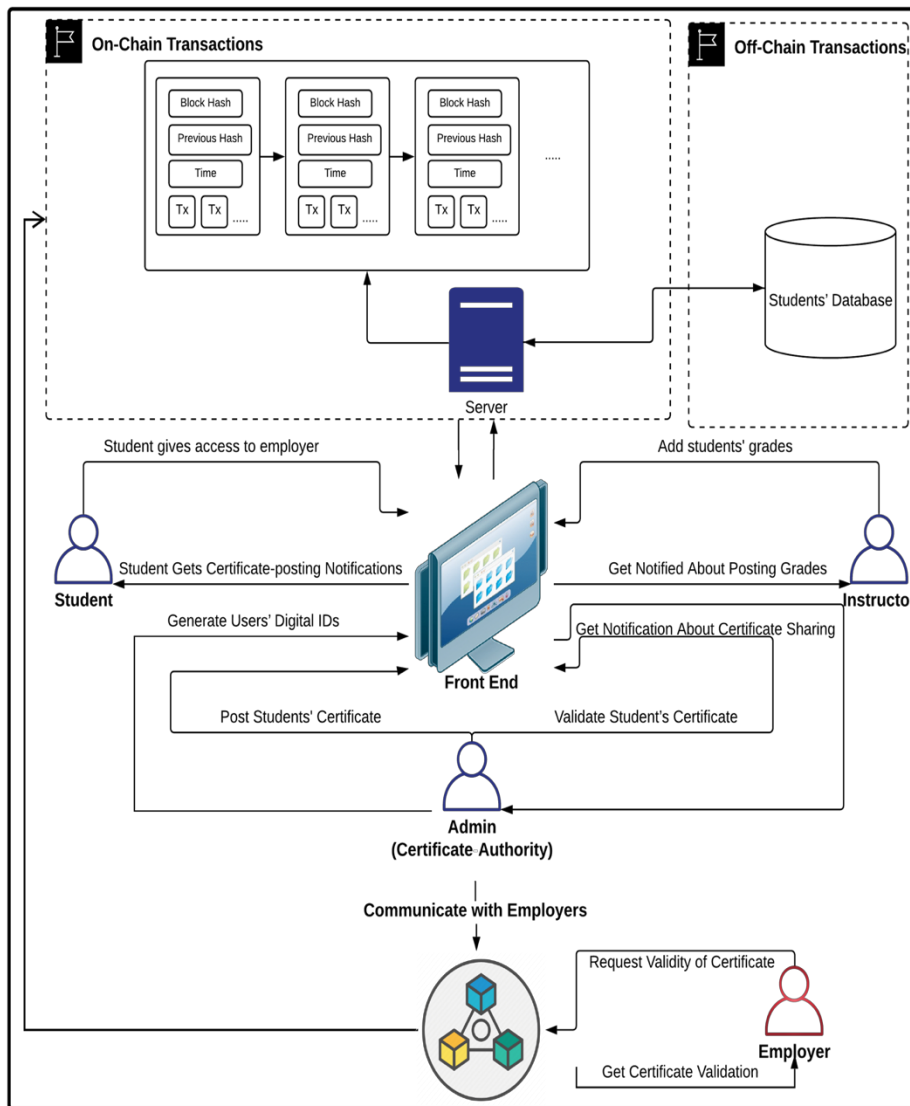


Figure 6-61 High-level conceptual infrastructure of DASC [108]

The DASC system gathers information on students' credentials and guarantees data integrity. The students can then give permission to share such information with external parties. In line with the conceptual infrastructure, it allows interactions between prospective employers and front-end systems controlled by system administrators giving the appropriate permissions. Since the system presents as a proof of concept of the smart campus architectural framework, see figure 6- 62, different case scenarios can be deployed in the system including posting a student's certificate, verifying it and sharing it with prospective employers.

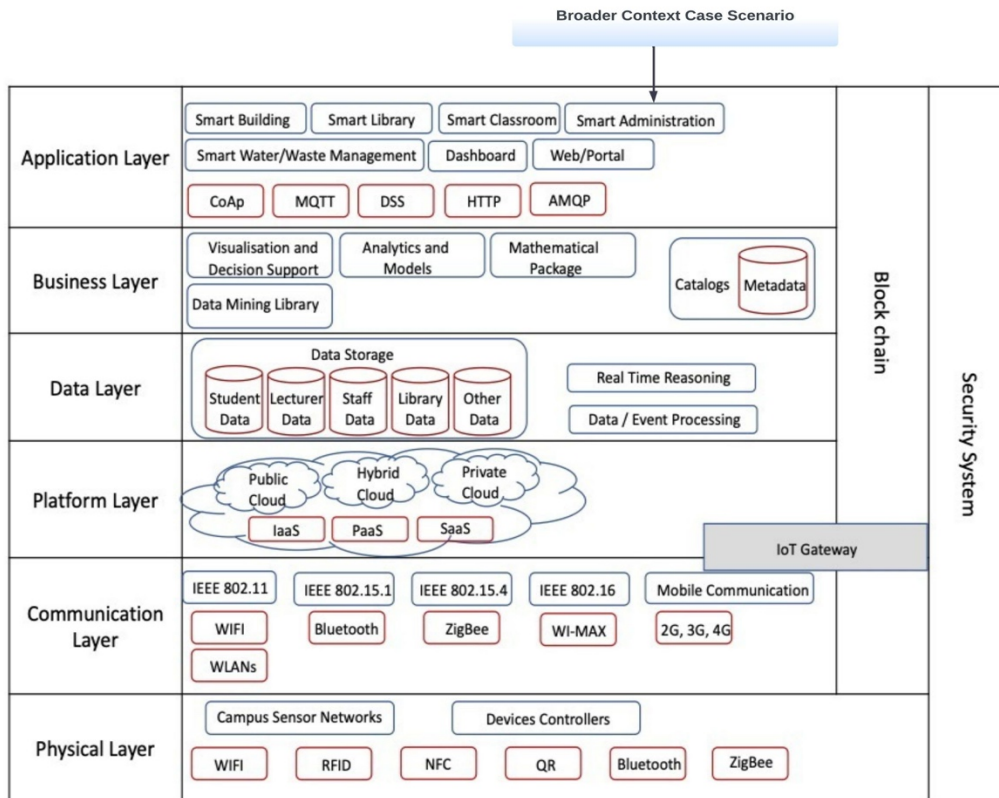


Figure 6-62 Broader context case scenario based the smart campus architectural framework

To sum up, the DASC system shows the applicability of the proposed smart campus architectural framework which combines blockchain and IoT technologies to provide more benefits by managing the problem associated with a current centralised IoT architecture particularly from a security perspective. Thus, the framework considered as guid base to develop various campus services, such as smart education, smart building and smart parking. Next, overall discussion will be provided.

6.6. Discussion

This novel smart-campus framework architecture was developed using the advantages of blockchain technology to provide benefits such as immutability, transparency, and data provenance by managing the problems associated with a centralised Internet of Things (IoT) architecture, especially from a security perspective. In addition, the framework provides availability, scalability, and flexibility by utilising an emerging IPFS platform. The system advantages can be summarised in the following ways:

- **Robust:** The system is totally decentralised as both Hyperledger Fabric blockchain and IPFS platforms are storing data in a distributed fashion and do not require a centralised third party. In addition, the system avoids data redundancy. Data redundancy is considered a challenge that faces many smart-system environments and connected devices [245], leading to computational overhead and heavy communication on the network as well as an increased transactional workload on the user's device [246], which results in an inefficient use of resources. In the IPFS platform, however, that data is segmented to be stored in the network and the platform does not use a redundancy mechanism. In frequent-access data mode, the system automatically stores the data on the server cache and removes the data that is not being accessed frequently, thereby managing the number of copies of the data content.
- **Efficient:** The system provides flexibility in managing data storage and security in decentralised access, and it ensures data availability. Clients can retrieve data and perform queries from their local network in the smart campus, rather than data traveling to the headquarter traffic. Thus, the system can use and combine network resources, making storage more automated and efficient. Joining the system requires having a shared key; this enables the user to access the private IPFS network and to have a role in accessing the blockchain network and configured genesis file.
- **Immutability:** Since every transaction in the system is stored in the ledger, each node has a full copy of it and each block is connected cryptographically to the previous block: all of this provides immutability, traceability, and security to the system as well making it hard to sabotage the stored data. In addition, the data that is stored off-chain is uniquely identified, so that any change to the data leads to a change in its hash, even if the change is only one character. The generated hash value from IPFS will be sealed using Hyperledger Fabric, onto its immutable ledger. Therefore, the system provides data integrity and immutability, and it is a suitable solution to prevent data tampering.
- **Scalability:** With the use of distributed files, the system is able to scale its storage capability without interrupting its services especially for large organisations such as smart campuses that are spread over different locations, which would increase the number of IoT devices needed to help to process all commands and transport decisions efficiently.

- **Availability:** The system deploys blockchain and IPFS as decentralised technologies that do not rely on central instance, thereby efficiently solving the problem of the single point of failure – where the whole system can be down and unavailable.
- **Integrity:** The system guarantees data integrity. Data that is collected from network gateways has to be encrypted to store in the decentralised structure. In addition, a signed transaction cannot be modified or have its content changed by any entities. The connected peers do not have the right to access policies in the system or change the smart contract agreement. Thus, in the smart-campus framework, sensitive data – such as student information – can be shared between authorised peers without any alteration.
- **Privacy:** The system-access control approach guarantees data ownership and data privacy of individuals. The use of the Hyperledger Fabric blockchain allows authenticated entities to access the network under predefined security policies and guidelines. Thus, malicious or unauthorised access is blocked, and is prevented from accessing the system. In addition, the use of the consensus algorithm prevents any illegal transactions from being validated or attached to the blockchain. Therefore, the system can monitor all transactions and detect any modification to the data.

Overall, this analysis shows that the novel smart campus framework can safeguard its data and preserve itself against potential threats and attacks. In addition, the system has many useful features, such as security, privacy, scalability, availability, and integrity, all of which can be beneficial to develop smart campus services and applications. Thus, the scheme can deliver a promising solution for enhancing current smart-campus frameworks and applications.

6.7. Summary

This chapter presented a real implementation of different case scenarios on smart campuses – in particular, a smart educational environment as a proof of concept of the smart campus architectural framework. To examine the performance of the proposed scheme, Hyperledger Fabric blockchain, with its entities, were deployed on Amazon EC2 instances, where different authorised peers could interact with the system through a developed web application. In addition, using P2P private IPFS technology as off-chain storage has integrated the blockchain to provide a totally decentralised structure for storing and sharing data. The results of the implementation show that the framework achieved the quality requirements for a smart

education system, as discussed in section 4.1, delivered in a lightweight and reliable manner. In addition, the chapter presented the applicability of a future comprehensive framework for a smart campus in a broader context case scenario, aiming to develop a student certification framework with higher education in Saudi Arabia. Furthermore, the chapter discussed extensive analysis and evaluations on several design aspects of the proposed smart-campus framework, which this study believes is an efficient solution and a step towards improving current smart-campus frameworks and applications.

Chapter 7 Conclusion and Future Research

This chapter provides an overview of the whole study, focusing on how to develop a smart campus, and summarises its results and achievements. In addition, this chapter highlights the study's limitations and the directions of the further research.

7.1. The Context of the Research

The focus on smart campuses has been fuelled by the advancement of advanced technologies such as IoT, blockchain and machine learning, which enable data to be captured at the level of the infrastructure, generating valuable insights for the stakeholders. However, because of centralised architecture, there are still security and privacy issues in the current smart campus system that have been discussed in Chapter 2.

In the last decade, blockchain technology research has become a growing trend in computer science, with increasing attention from various researchers and organisations. Since 2008, it has taken a place among the top five technology trends and is considered to be the next revolution in technology as it provides solutions to issues related to classical centralised architecture, particularly regarding security and privacy. Blockchain properties – including decentralisation, anonymity, resiliency, autonomous control, and support for integrity – put it in the lead as a suitably advanced technology to apply to the smart campus architectural framework. The aims and objectives of this study developed from this context.

7.2. Research Questions

the main aim of this research was to propose and evaluate a comprehensive framework for a smart campus implementation. To achieve this objective of this study, it was necessary to answer the following research main questions:

- **What technologies, including hardware and software, contribute to the building of a highly technological smart campus?** Current state-of-the-art research relevant to smart campus has been conducted in order to gain knowledge on existing smart technologies and intelligent software/hardware systems in use in different campus sectors. In addition, recent smart campus frameworks and architecture have been analysed to determine their limitations, as shown in Chapter 2.
- **How can the smart campus technologies be characterised?** A novel, holistic smart campus architectural framework (illustrated in Figure 3-1) was designed, implemented, and evaluated. The framework emerged the advantages of IoT and blockchain technology to eliminate current centralised issues. That is, these technologies can collect and aggregate data from various areas of the campus while increasing data security and providing a better service to enhance user experience. Integrating blockchain technology into education institutions is still in its early stages and needs more research. Security requirements for the proposed smart campus architectural framework were analysed, focusing on issues of authorisation, privacy, confidentiality, integrity, and availability.
- **What methods can be used to develop a comprehensive framework for a smart campus implementation?** A systematic analysis was conducted to determine which blockchain platform and consensus algorithms are suitable, dependant on the discussion of the quality requirements for blockchain-based smart education environments. As shown in Chapter 4, the Hyperledger Fabric platform, with Raft consensus algorithm, is an ideal blockchain for storing and sharing sensitive data such as that of educational institutes, maintaining privacy and confidentiality within the network. In addition, IPFS, as a form of distributed off-chain storage, was considered for use alongside Hyperledger Fabric to increase the blockchain scalability, as in Chapter 5. Thus, the designed model is immutable and content addressed, as well as ensuring the security, privacy, and integrity of the data. Different case scenarios were designed and implemented as proof of concept to facilitate a smart educational environment using Hyperledger Fabric 2.3.2 and a private IPFS network, as in Chapter 6.
- **What are the implications on the applicability of the comprehensive framework for a smart campus in the future?** A series of tests were carried out, as in chapter 6, and another experiment was conducted for a broader context (student certification

scenario). As a result, the study showed the efficiency and the applicability of the novel, holistic, comprehensive framework for a smart campus.

7.3. The Aims and Objectives

The main goal of this study was to propose and evaluate a comprehensive framework for a smart campus implementation since no comprehensive guiding framework has been developed for emerging IoT and blockchain technologies deployment in this field, specifically in relation to security and privacy issues and the mitigation of known problems with IoT and blockchain in existing applications.

To attain the study's main goal, it was necessary to separate it into several sections:

- Determining appropriate technologies that can be utilised for the development of the framework by gaining knowledge on related research work and understanding smart campuses' principles and concepts.
- Extracting domain knowledge by investigating the main features of emerging IoT and blockchain technologies.
- Gaining new insights by analysing the validity of distributing smart campus applications and systems.
- Choosing different case scenario to be implemented and evaluated for the applicability of the holistic architectural framework.

7.4. Contributions to Knowledge

The main achievements of this research are summarised as follows:

- 1 The research has presented in the literature review provides an analysis of the current state of the art relevant to smart campuses and the use of existing smart technologies in different campus sectors, such as education, building management, waste management, energy management, water management, transportation, and security.
- 2 This research has analysed the best technologies including consensus algorithms, blockchain platforms and off-chain storages to use in different case scenarios as proof of concept.

- 3 This research has developed a novel, comprehensive, smart campus architectural framework, using the advantages of blockchain technology to handle issues related to a centralised IoT architecture.
- 4 This research has developed a systematic analysis method to determine which blockchain platform and consensus algorithms are suitable for blockchain-based smart education environment.
- 5 This research has evaluated the framework by assessing performance and security requirements, which has shown that the system is lightweight since it avoids high network latency and low throughput.
- 6 This research has tested the future applicability of the comprehensive framework for a smart campus in a broader context (student certification scenario).

7.5. Summary of the thesis

In this section, the summary of this thesis is as the follows:

- Current state-of-the-art research relevant to smart campus has been conducted in order to gain knowledge on existing smart technologies and intelligent software/hardware systems in use in different campus sectors. In addition, recent smart campus frameworks and architecture have been analysed to determine their limitations, as shown in Chapter 2.
- A novel, holistic smart campus architectural framework (illustrated in Figure 3-1) was designed, implemented, and evaluated. The framework emerged the advantages of IoT and blockchain technology to eliminate current centralised issues. That is, these technologies can collect and aggregate data from various areas of the campus while increasing data security and providing a better service to enhance user experience.
- Integrating blockchain technology into education institutions is still in its early stages and needs more research. Security requirements for the proposed smart campus architectural framework from were analysed, focusing on issues of authorisation, privacy, confidentiality, integrity, and availability.
- A systematic analysis was conducted to determine which blockchain platform and consensus algorithms are suitable, dependant on the discussion of the quality requirements for blockchain-based smart education environments. As shown in Chapter

4, the Hyperledger Fabric platform, with Raft consensus algorithm, is an ideal blockchain for storing and sharing sensitive data such as that of educational institutes, maintaining privacy and confidentiality within the network.

- IPFS, as a form of distributed off-chain storage, was considered for use alongside Hyperledger Fabric to increase the blockchain scalability, as in Chapter 5. Thus, the designed model is immutable and content addressed, as well as ensuring the security, privacy, and integrity of the data.
- Different case scenarios were designed and implemented as proof of concept to facilitate a smart educational environment using Hyperledger Fabric 2.3.2 and a private IPFS network, as in Chapter 6.
- A series of tests were carried out, as in chapter 6, and another experiment was conducted for a broader context (student certification scenario). As a result, the study showed the efficiency and the applicability of the novel, holistic, comprehensive framework for a smart campus.

7.6. Research Limitations and Directions of Future Work

Blockchain technology is a fairly new revolution and has shown successful results and adaptations in the area of digital cryptocurrencies. This study therefore took a step further and used blockchain beyond the financial field. The research provides a solid basis for further study, particularly in decentralised system domains, and open new constitutes and avenues for future research. With the development of the holistic comprehensive framework for a smart campus and the achievement of the main objectives, the study concluded that blockchain technology with its characteristics is a great tool that can be used to decentralise campus systems, to eliminate IoT centralisation issues and to improve security and privacy. Since the research presented proof of concept for the framework, further study could include several developments in the systems of different areas of smart campuses, or in business systems in the development of smart city environments generally, taking into account their business needs and quality requirements.

Blockchain technology is still in its early stages particularly in developing smart campuses systems where most of proposed applications are still not being implemented. Therefore, the

research systematic analyses were focussed on the current well-known blockchain platforms and consensus algorithms, which is a mechanism utilised by various platforms of blockchain networks to ensure the consistency, availability and integrity of stored data across geographically distributed peers. There are several different novel blockchain platforms and consensus algorithms that was developed by researchers. However, this research focused on and analysed existing adopted consensus algorithms and open sources blockchain platforms which considered a research limitation. Novel ones were beyond the scope of this research and may provide the subject of another possible future research work.

Integrating IoT and blockchain technology in the proposed architectural framework brought various benefits in increased security and privacy of educational data by storing it in a distributed and encrypted form. For a next step, integrating artificial intelligence (AI) technology such as neural networks, machine learning and deep learning into the architectural framework would be a great direction for further study. Recently, AI has shown a significant effect on among different areas, producing 'smart' activities and models. The three technologies of IoT, blockchain, and AI deliver unlimited, innovative possibilities on the future, offering benefits such as data-gathering through IoT devices, smart data analysis and advanced decision-making algorithms though AI, and time-stamped data storage through blockchain.

References

- [1] L. Kwok, "A vision for the development of i-campus," *Smart Learn. Environ.*, vol. 2, no. 1, p. 2, 2015.
- [2] W. Villegas-Ch, X. Palacios-Pacheco, and S. Luján-Mora, "Application of a smart city model to a traditional university campus with a big data architecture: A sustainable smart campus," *Sustain.*, vol. 11, no. 10, 2019.
- [3] R. Szabo et al., "Framework for smart city applications based on participatory sensing," in *4th IEEE International Conference on Cognitive Infocommunications, CogInfoCom 2013 - Proceedings*, 2013, pp. 295–300.
- [4] A. Caragliu, C. Del Bo, and P. Nijkamp, "Smart Cities in Europe Smart Cities in Europe," *3rd Cent. Eur. Conf. Reg. Sci.*, vol. 0732, no. November, pp. 1–15, 2015.
- [5] C. Perera, C. H. Liu, S. Jayawardena, and M. Chen, "A Survey on Internet of Things from Industrial Market Perspective," *IEEE Access*, vol. 2, pp. 1660–1679, 2015.
- [6] M. I. Pramanik, R. Y. K. Lau, H. Demirkan, and M. A. K. Azad, "Smart health: Big data enabled health paradigm within smart cities," *Expert Systems with Applications*. 2017.
- [7] L. Catarinucci et al., "An IoT-Aware Architecture for Smart Healthcare Systems," *IEEE Internet Things J.*, 2015.
- [8] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, "Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare," *Futur. Gener. Comput. Syst.*, 2018.
- [9] S. Amendola, R. Lodato, S. Manzari, C. Occhiuzzi, and G. Marrocco, "RFID technology for IoT-based personal healthcare in smart spaces," *IEEE Internet Things J.*, 2014.
- [10] E. M. Tachizawa, M. J. Alvarez-Gil, and M. J. Montes-Sancho, "How 'smart cities' will change supply chain management," *Supply Chain Manag.*, 2015.
- [11] J. Lukić, M. Radenković, M. Despotović-Zrakić, A. Labus, and Z. Bogdanović, "Supply chain intelligence for electricity markets: A smart grid perspective," *Inf. Syst. Front.*, 2017.
- [12] B. Ghazal, K. Elkhatib, K. Chahine, and M. Kherfan, "Smart traffic light control system," in *2016 3rd International Conference on Electrical, Electronics, Computer Engineering and their Applications, EECEA 2016*, 2016.
- [13] J. L. Galán-García, G. Aguilera-Venegas, and P. Rodríguez-Cielos, "An accelerated-time simulation for traffic flow in a smart city," *J. Comput. Appl. Math.*, 2014.
- [14] P. K. Nair, F. Ali, and C. L. Lim, "Interactive Technology and Smart Education Article information :," *Interact. Technol. Smart Educ.*, 2015.
- [15] A. Alelaiwi, A. Alghamdi, M. Shorfuzzaman, M. Rawashdeh, M. S. Hossain, and G. Muhammad, "Enhanced engineering education using smart class environment," *Comput. Human Behav.*, 2015.

- [16] M. S. Ibrahim, A. Z. A. Razak, and H. B. Kenayathulla, "Smart Principals and Smart Schools," *Procedia - Soc. Behav. Sci.*, 2013.
- [17] W. Muhamad, N. B. Kurniawan, S. Suhardi, and S. Yazid, "Smart campus features, technologies, and applications: A systematic literature review," in *2017 International Conference on Information Technology Systems and Innovation, ICITSI 2017 - Proceedings*, 2018.
- [18] X. Dong, X. Kong, F. Zhang, Z. Chen, and J. Kang, "OnCampus: a mobile platform towards a smart campus Background," *Springerplus*, vol. 5, 2016.
- [19] K. Peffers, C. Gengler, T. Tuunanen, and M. Rossi, "The Design Science Research Process: a Model for Producing and Presenting Information System Research," *Proc. first Int. Conf. Des. Sci. Res. Inf. Syst. Technol.*, no. May 2014, 2006.
- [20] S. Gregor and A. R. Hevner, "Positioning and presenting design science research for maximum impact," *MIS Quarterly: Management Information Systems*, vol. 37, no. 2, 2013.
- [21] P. Offermann, O. Levina, M. Schönherr, and U. Bub, "Outline of a design science research process," in *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, DESRIST '09*, 2009.
- [22] N. Ahern and D. M. Wink, "Virtual learning environments: Second life," *Nurse Educ.*, 2010.
- [23] A. Alam and S. Ullah, "Adaptive 3D-Virtual Learning Environments: From Students' Learning Perspective," in *Proceedings - 14th International Conference on Frontiers of Information Technology, FIT 2016*, 2017.
- [24] H. Komaki, S. Shimazaki, K. Sakakibara, and T. Matsumoto, "Interactive optimization techniques based on a column generation model for timetabling problems of university makeup courses," in *2015 IEEE 8th International Workshop on Computational Intelligence and Applications, IWCI 2015 - Proceedings*, 2016.
- [25] R. Mei, J. Guan, and B. Li, "University course timetable system design and implementation based on mathematical model," in *2010 The 2nd International Conference on Computer and Automation Engineering, ICCAE 2010*, 2010.
- [26] A. Abuarqoub et al., "A Survey on Internet of Thing Enabled Smart Campus Applications," *Proc. Int. Conf. Futur. Networks Distrib. Syst. - ICFNDS '17*, pp. 1–7, 2017.
- [27] Y. Khamayseh, W. Mardini, S. Aljawarneh, and M. B. Yassein, "Integration of Wireless Technologies in Smart University Campus Environment," *Int. J. Inf. Commun. Technol. Educ.*, vol. 11, no. 1, pp. 60–74, 2015.
- [28] Y. Atif, S. S. Mathew, and A. Lakas, "Building a smart campus to support ubiquitous learning," *J. Ambient Intell. Humaniz. Comput.*, vol. 6, no. 2, pp. 223–238, 2015.
- [29] Y. Chen, R. Zhang, X. Shang, and S. Zhang, "An intelligent campus space model based on the service encapsulation," in *LISS 2012 - Proceedings of 2nd International Conference on Logistics, Informatics and Service Science*, 2013, pp. 919–923.
- [30] Y. Chen, X. Li, Y. Wang, and L. Gao, "The design and implementation of intelligent campus security tracking system based on RFID and ZigBee," in *2011 2nd International Conference on Mechanic Automation and Control Engineering, MACE 2011 - Proceedings*, 2011, pp. 1749–1752.

- [31] J. W. P. Ng, N. Azarmi, M. Leida, F. Saffre, A. Afzal, and P. D. Yoo, "The intelligent campus (iCampus): End-to-end learning lifecycle of a knowledge ecosystem," in *Proceedings - 2010 6th International Conference on Intelligent Environments, IE 2010, 2010*, pp. 332–337.
- [32] P. M. Jackson, "Intelligent campus," in *SPCA 2006: 2006 First International Symposium on Pervasive Computing and Applications, Proceedings, 2007*, p. 3.
- [33] B. Hirsch and J. W. P. Ng, "Education beyond the cloud: Anytime-anywhere learning in a smart campus environment," *2011 Int. Conf. Internet Technol. Secur. Trans.*, no. December, pp. 718–723, 2011.
- [34] Y. Atif and S. Mathew, "A social web of things approach to a smart campus model," in *Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCom 2013, 2013*, pp. 349–354.
- [35] Y. L. Liu, W. H. Zhang, and P. Dong, "Research on the Construction of Smart Campus Based on the Internet of Things and Cloud Computing," *Appl. Mech. Mater.*, 2014.
- [36] A. Adamkó and L. Kollár, "A system model and applications for intelligent campuses," in *INES 2014 - IEEE 18th International Conference on Intelligent Engineering Systems, Proceedings, 2014*, pp. 193–198.
- [37] A. Boran, I. Bedini, C. J. Matheus, P. F. Patel-Schneider, and J. Keeney, "A smart campus prototype for demonstrating the semantic integration of heterogeneous data," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6902 LNCS, pp. 238–243.
- [38] Z. Yu, Y. Liang, B. Xu, Y. Yang, and B. Guo, "Towards a smart campus with mobile social networking," in *Proceedings - 2011 IEEE International Conferences on Internet of Things and Cyber, Physical and Social Computing, iThings/CPSCom 2011, 2011*.
- [39] E. De Angelis, A. L. C. Ciribini, L. C. Tagliabue, and M. Paneroni, "The Brescia Smart Campus Demonstrator. Renovation toward a zero Energy Classroom Building," in *Procedia Engineering, 2015*, vol. 118, pp. 735–743.
- [40] D. Kolokotsa et al., "Development of a web based energy management system for University Campuses: The CAMP-IT platform," *Energy Build.*, vol. 123, pp. 119–135, 2016.
- [41] Y. Han and K. Xia, "Data preprocessing method based on user characteristic of interests for web log mining," in *Proceedings - 2014 4th International Conference on Instrumentation and Measurement, Computer, Communication and Control, IMCCC 2014, 2014*, pp. 867–872.
- [42] W. Kuang and N. Luo, "User interests mining based on topic map," in *Proceedings - 2010 7th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2010, 2010*, vol. 5, pp. 2399–2402.
- [43] A. I. Amr, S. Kamel, G. El Gohary, and J. Hamhaber, "Water as an Ecological Factor for a Sustainable Campus Landscape," *Procedia - Soc. Behav. Sci.*, vol. 216, pp. 181–193, 2016.
- [44] G. B. Shi, "The design of campus monitoring and managing system for watersaving based on webgis," *Proc. - 2017 IEEE Int. Conf. Internet Things, IEEE Green Comput.*

- Commun. IEEE Cyber, Phys. Soc. Comput. IEEE Smart Data, iThings-GreenCom-CPSCom-SmartData 2017, vol. 2018-Janua, pp. 951–954, 2018.
- [45] V. D. Kudva et al., “Towards a Real-Time Campus-Scale Water Balance Monitoring System,” in Proceedings of the IEEE International Conference on VLSI Design, 2015, vol. 2015-Febru, no. February, pp. 87–92.
- [46] P. Verma et al., “Towards an IoT based water management system for a campus,” in 2015 IEEE 1st International Smart Cities Conference, ISC2 2015, 2015.
- [47] A. Alghamdi and S. Shetty, “Survey toward a smart campus using the internet of things,” in Proceedings - 2016 IEEE 4th International Conference on Future Internet of Things and Cloud, FiCloud 2016, 2016, pp. 235–239.
- [48] M. J. Mudumbe and A. M. Abu-Mahfouz, “Smart water meter system for user-centric consumption measurement,” in Proceeding - 2015 IEEE International Conference on Industrial Informatics, INDIN 2015, 2015, pp. 993–998.
- [49] S. Goenka and R. S. Mangrulkar, “Robust Waste Collection: Exploiting IOT Potentiality in Smart Cities,” *i-Manager’s J. Softw. Eng.*, vol. 11, no. 3, pp. 10–18, 2017.
- [50] F. Foliato, Y. S. Low, and W. L. Yeow, “Smartbin: Smart waste management system,” in 2015 IEEE 10th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2015, 2015.
- [51] K. Ebrahimi, L. North, and J. Yan, “GIS applications in developing zero-waste strategies at a mid-size American university,” in International Conference on Geoinformatics, 2017, vol. 2017-Augus.
- [52] R. W. Ahmad, K. Salah, R. Jayaraman, I. Yaqoob, and M. Omar, “Blockchain for Waste Management in Smart Cities: A Survey,” *IEEE Access*, vol. 9, 2021.
- [53] S. A. Saad, A. A. B. Hisham, M. H. I. Ishak, M. H. M. Fauzi, M. A. Baharudin, and N. H. Idris, “Real-time on-campus public transportation monitoring system,” in Proceedings - 2018 IEEE 14th International Colloquium on Signal Processing and its Application, CSPA 2018, 2018.
- [54] M. Ramadan, M. Al-Khedher, and S. Al-Kheder, “Intelligent anti-theft and tracking system for automobiles,” *Int. J. Mach. Learn. Comput.*, 2012.
- [55] S. Priya, B. Prabhavathi, P. Shanmuga Priya, B. Shanthini, and U. Scholar, “An Android Application for Tracking College Bus Using Google Map,” *Int. J. Comput. Sci. Eng. Commun.*, 2015.
- [56] M. P. Suresh Mane and P. V. Khairnar, “Analysis of Bus Tracking System Using Gps on Smart Phones,” *IOSR J. Comput. Eng.*, 2014.
- [57] M. Wazid, B. Bera, A. K. Das, S. P. Mohanty, and M. Jo, “Fortifying Smart Transportation Security Through Public Blockchain,” *IEEE Internet Things J.*, vol. 9, no. 17, 2022.
- [58] J. E. Ferreira, J. A. Visintin, J. Okamoto, and C. Pu, “Smart services: A case study on smarter public safety by a mobile app for University of São Paulo,” 2017 IEEE SmartWorld Ubiquitous Intell. Comput. Adv. Trust. Comput. Scalable Comput. Commun. Cloud Big Data Comput. Internet People Smart City Innov. SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI 2017 - , pp. 1–5, 2018.
- [59] Y. Wang et al., “A smart campus internet of things framework,” in 2017 IEEE 8th

- Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2017, 2018.
- [60] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," *Inf. Syst. Front.*, vol. 17, no. 2, pp. 243–259, 2015.
- [61] N. Mishra, P. Singhal, and S. Kundu, "Application of IoT products in smart cities of India," in *Proceedings of the 2020 9th International Conference on System Modeling and Advancement in Research Trends, SMART 2020*, 2020, pp. 155–157.
- [62] A. Nayyar, V. Puri, and D.-N. Le, "Internet of Nano Things (IoNT): Next Evolutionary Step in Nanotechnology," *Nanosci. Nanotechnol.*, vol. 7, no. 1, pp. 4–8, 2017.
- [63] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, 2015.
- [64] K. Zhao and L. Ge, "A survey on the internet of things security," in *Proceedings - 9th International Conference on Computational Intelligence and Security, CIS 2013*, 2013, pp. 663–667.
- [65] D. Pennino, M. Pizzonia, A. Vitaletti, and M. Zecchini, "Blockchain as IoT Economy Enabler: A Review of Architectural Aspects," *J. Sens. Actuator Networks*, vol. 11, no. 2, p. 20, 2022.
- [66] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with IoT . Challenges and opportunities," vol. 88, pp. 173–190, 2018.
- [67] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," *Proc. - 2017 IEEE 6th Int. Congr. Big Data, BigData Congr. 2017*, pp. 557–564, 2017.
- [68] F. Vidal, F. Gouveia, and C. Soares, "Analysis of blockchain technology for higher education," *Proc. - 2019 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2019*, pp. 28–33, 2019.
- [69] K. Panetta and Gartner, "5 Trends Emerge in the Gartner Hype Cycle for Emerging Technologies, 2018 - Smarter With Gartner," Gartner, 2018. .
- [70] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain Technology Overview," *arXiv*. 2019.
- [71] K. Sultan, U. Ruhi, and R. Lakhani, "Conceptualizing blockchains: Characteristics & applications," in *Proceedings of the 11th IADIS International Conference Information Systems 2018, IS 2018*, 2018.
- [72] A. Narayanan and J. Clark, "Bitcoin's academic pedigree," *Communications of the ACM*. 2017.
- [73] Coinbase, "Bitcoin Price," Coinbase, 2018.
- [74] W. A. Kaal and M. Dell'Erba, "Blockchain Innovation in Private Investment Funds - A Comparative Analysis of the United States and Europe," *SSRN Electron. J.*, 2018.
- [75] V. Buterin, "Ethereum White Paper," Ethereum, 2014.
- [76] V. J. Morkunas, J. Paschen, and E. Boon, "How blockchain technologies impact your business model," *Bus. Horiz.*, vol. 2018, no. 2018, 2019.

- [77] O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT," *IEEE Internet Things J.*, 2018.
- [78] N. Parlante, "Linked List Basics," *Stanford CS Educ. Libr.*, 2002.
- [79] U. Gandhi, "a Review Towards Various Hash Algorithms and Their Comparative Analysis," *Int. Res. J. Eng. Technol.*, 2017.
- [80] Y. Hassanzadeh-Nazarabadi, A. K p c , and  .  zkasap, "LightChain: Scalable DHT-Based Blockchain," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, 2021.
- [81] P. P. Pittalia, "A Comparative Study of Hash Algorithms in Cryptography," *Int. J. Comput. Sci. Mob. Comput.*, 2019.
- [82] S. Verma, M. T. Scholar, and G. S. P. Head, "A Survey of Cryptographic Hash Algorithms and Issues," vol. 1, pp. 17–20, 2015.
- [83] F. Chabaud and A. Joux, "Differential collisions in SHA-0," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1998.
- [84] P. Jones, "US secure hash algorithm 1 (SHA1) RFC 3174," RFC 3174, 2001.
- [85] N. Sklavos and O. Koufopavlou, "Implementation of the SHA-2 hash family standard using FPGAs," *J. Supercomput.*, 2005.
- [86] R. K. Ibrahim, R. A. J. Kadhim, and A. S. H. Alkhalid, "Incorporating SHA-2 256 with OFB to realize a novel encryption method," in *2015 World Symposium on Computer Networks and Information Security, WSCNIS 2015*, 2015.
- [87] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013.
- [88] R. Martino and A. Cilaro, "Designing a SHA-256 processor for blockchain-based IoT applications," *Internet of Things*, 2020.
- [89] R. Grinberg, "Bitcoin: An Innovative Alternative Digital Currency," *Hast. Sci. Technol. Law J.*, 2011.
- [90] C. Cachin and M. Vukoli , "Blockchain consensus protocols in the wild," in *Leibniz International Proceedings in Informatics, LIPIcs*, 2017.
- [91] D. Tosh, S. Shetty, P. Foytik, C. Kamhoua, and L. Njilla, "CloudPoS: A Proof-of-Stake Consensus Design for Blockchain Integrated Cloud," in *IEEE International Conference on Cloud Computing, CLOUD*, 2018.
- [92] D. Burkhardt, M. Werling, and H. Lasi, "Distributed Ledger," in *2018 IEEE International Conference on Engineering, Technology and Innovation, ICE/ITMC 2018 - Proceedings*, 2018.
- [93] L. Ismail, H. Hameed, M. Aishamsi, M. Aihammadi, and N. Aidhanhani, "Towards a blockchain deployment at UAE University: Performance evaluation and blockchain taxonomy," in *ACM International Conference Proceeding Series*, 2019, vol. Part F1481, pp. 30–38.
- [94] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*. 2016.
- [95] M. Suvitha and R. Subha, "A Survey on Smart Contract Platforms and Features," in

- 2021 7th International Conference on Advanced Computing and Communication Systems, ICACCS 2021, 2021, pp. 1536–1539.
- [96] F. Casino, T. K. Dasaklis, and C. Patsakis, “A systematic literature review of blockchain-based applications: Current status, classification and open issues,” *Telematics and Informatics*. 2019.
- [97] U. Bodkhe et al., “Blockchain for Industry 4.0: A comprehensive review,” *IEEE Access*, 2020.
- [98] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System(HP),” Consulted, 2008.
- [99] The Economist, “The great chain of being sure about things,” *Econ.*, 2015.
- [100] B. Adam et al., “Enabling Blockchain Innovations with Pegged Sidechains, <https://blockstream.com/sidechains.pdf>,” URL <http://www.>, 2014.
- [101] V. Buterin, “Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform,” *Ethereum*, 2013.
- [102] A. Hughes, A. Park, J. Kietzmann, and C. Archer-Brown, “Beyond Bitcoin: What blockchain and distributed ledger technologies mean for firms,” *Bus. Horiz.*, 2019.
- [103] G. G. Dagher, J. Mohler, M. Milojkovic, and P. B. Marella, “Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology,” *Sustain. Cities Soc.*, 2018.
- [104] S. Huh, S. Cho, and S. Kim, “Managing IoT devices using blockchain platform,” in *International Conference on Advanced Communication Technology, ICACT*, 2017.
- [105] Hyperledger, “Hyperledger Whitepaper,” Publ. <https://wiki.hyperledger.org/groups/whitepaper/whitepaper-wg>, 2016.
- [106] P. Mukherjee and C. Pradhan, “Blockchain 1.0 to blockchain 4.0—The evolutionary transformation of blockchain technology,” in *Intelligent Systems Reference Library*, vol. 203, 2021, pp. 29–49.
- [107] M. Alkhamash, N. Beloff, and M. White, “An Internet of Things and Blockchain Based Smart Campus Architecture,” in *Intelligent Computing*, 2020, pp. 467–486.
- [108] M. Alkhamash, M. Alshahrani, N. Beloff, and M. White, “Revolutionising the Approach to Smart Campus Architecture through IoT and Blockchain Technologies,” in *TTBT2021*, Springer International Publishing, 2021.
- [109] J. Kaur and O. Jyotsna, “Blockchain Technology in Education Sector: A Review,” *JAC A J. Compos. Theory*, vol. 8, no. 4, 2020.
- [110] O. S. Saleh, O. Ghazali, and M. E. Rana, “Blockchain based framework for educational certificates verification,” *Journal of Critical Reviews*, vol. 7, no. 3. 2020.
- [111] S. Murugesan and M. B. Lakshminarasaiyah, “A survey on blockchain-based student certificate management system,” *ACM Int. Conf. Proceeding Ser.*, pp. 44–50, 2021.
- [112] Broggi, Lilly, and Duquette, “Building the first blockchain university,” *Woolf Dev. Ltd*, 2018.
- [113] M. Holbl, A. Kamisalic, M. Turkanovic, M. Kompara, B. Podgorelec, and M. Hericko, “EduCTX: An Ecosystem for Managing Digital Micro-Credentials,” in *2018 28th EAEEIE Annual Conference, EAEEIE 2018*, 2018.

- [114] A. Rachmat and Albarda, "Design of Distributed Academic-record System Based on Blockchain," in *Proceeding - 2019 International Conference on ICT for Smart Society: Innovation and Transformation Toward Smart Region, ICISS 2019*, 2019, pp. 1–7.
- [115] X. Liu, "A smart book management system based on Blockchain platform," in *Proceedings - 2019 International Conference on Communications, Information System, and Computer Engineering, CISCE 2019*, 2019.
- [116] S. Sharma and R. S. Batth, "Blockchain Technology for Higher Education Sytem: A Mirror Review," in *Proceedings of International Conference on Intelligent Engineering and Management, ICIEM 2020*, 2020.
- [117] W. Y. Chiu, W. Meng, and W. Li, "LibBlock - Towards Decentralized Library System based on Blockchain and IPFS," in *2021 18th International Conference on Privacy, Security and Trust, PST 2021*, 2021.
- [118] J. Zeng, X. Dai, J. Xiao, W. Yang, W. Hao, and H. Jin, "BookChain: Library-free book sharing based on blockchain technology," in *Proceedings - 2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN 2019*, 2019.
- [119] S. G. Education and Sony Global Education, "Sony Develops System for Authentication, Sharing, and Rights Management Using Blockchain Technology," Sony Corporation Sony Global Education, 2017.
- [120] B. Hameed et al., "A review of Blockchain based educational projects," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 10, 2019.
- [121] R. Arenas and P. Fernandez, "CredenceLedger: A Permissioned Blockchain for Verifiable Academic Credentials," in *2018 IEEE International Conference on Engineering, Technology and Innovation, ICE/ITMC 2018 - Proceedings*, 2018.
- [122] Y. Zou, T. Meng, P. Zhang, W. Zhang, and H. Li, "Focus on blockchain: A comprehensive survey on academic and application," *IEEE Access*, vol. 8, 2020.
- [123] A. Chowdhary, S. Agrawal, and B. Rudra, "Blockchain based Framework for Student Identity and Educational Certificate Verification," in *Proceedings of the 2nd International Conference on Electronics and Sustainable Communication Systems, ICESC 2021*, 2021.
- [124] B. Gipp, C. Breiting, N. Meuschke, and J. Beel, "CryptSubmit: Introducing Securely Timestamped Manuscript Submission and Peer Review Feedback Using the Blockchain," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, 2017.
- [125] T. Hepp, A. Schoenhals, C. Gondek, and B. Gipp, "OriginStamp: A blockchain-backed system for decentralized trusted timestamping," *IT - Inf. Technol.*, vol. 60, no. 5–6, pp. 273–281, 2018.
- [126] M. S. M. Pozi, G. Muruti, A. A. Bakar, A. Jatowt, and Y. Kawai, "Preserving Author Editing History Using Blockchain Technology," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, 2018, pp. 165–168.
- [127] Andi, R. Purba, and R. Yunis, "Application of Blockchain Technology to Prevent The Potential Of Plagiarism in Scientific Publication," in *Proceedings of 2019 4th International Conference on Informatics and Computing, ICIC 2019*, 2019.
- [128] T. Narendrakumar and A. S. Pillai, "Smart connected campus," in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies*,

- ICICICT 2017, 2018, vol. 2018-Janua, pp. 1591–1596.
- [129] I. Hossain, Di. Das, and M. G. Rashed, “Internet of Things Based Model for Smart Campus: Challenges and Limitations,” in 5th International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering, IC4ME2 2019, 2019.
- [130] H. Zhe, W. Xiaojun, and W. Qiang, “Research on campus information service platform based on locating system of campus wireless LAN,” Proc. - 2017 Int. Conf. Smart Grid Electr. Autom. ICSGEA 2017, vol. 2017-Janua, pp. 623–626, 2017.
- [131] H. Yan and H. Hu, “A study on association algorithm of smart campus mining platform based on big data,” in Proceedings - 2016 International Conference on Intelligent Transportation, Big Data and Smart City, ICITBS 2016, 2017, pp. 172–175.
- [132] H. Hu and H. Yan, “A study on discovery method of hot topics based on smart campus big data platform,” in Proceedings - 2016 International Conference on Intelligent Transportation, Big Data and Smart City, ICITBS 2016, 2017, pp. 176–179.
- [133] V. Agate, F. Concione, and P. Ferraro, “WiP: Smart services for an augmented campus,” in Proceedings - 2018 IEEE International Conference on Smart Computing, SMARTCOMP 2018, 2018, pp. 276–278.
- [134] J. Enqing, R. Peixiang, W. Huanjin, and S. Yanping, “Discussion on construction method of smart campus basic platform based on 3d geographic information technology,” in Proceedings - 2017 Chinese Automation Congress, CAC 2017, 2017, vol. 2017-Janua, pp. 7790–7794.
- [135] W. Lihong, “Research on the Construction of Smart Campus Social Platform Based on Hadoop,” in Proceedings - 2020 International Conference on Computer Engineering and Application, ICCEA 2020, 2020, pp. 214–217.
- [136] P. Agarwal, G. V. V. Ravi Kumar, and P. Agarwal, “IoT based framework for smart campus: COVID-19 readiness,” in Proceedings of the World Conference on Smart Trends in Systems, Security and Sustainability, WS4 2020, 2020, pp. 539–542.
- [137] O. Debauche, R. A. Abdelouahid, S. Mahmoudi, Y. Moussaoui, A. Marzak, and P. Manneback, “RevoCampus: A Distributed Open Source and Low-cost Smart Campus,” in 3rd International Conference on Advanced Communication Technologies and Networking, CommNet 2020, 2020.
- [138] M. A. Khan, “A survey of security issues for cloud computing,” Journal of Network and Computer Applications, vol. 71. pp. 11–29, 2016.
- [139] H. F. Atlam and G. B. Wills, “Intersections between IoT and distributed ledger,” Advances in Computers, 2019.
- [140] H. F. Atlam, E. El-Din Hemdan, A. Alenezi, M. O. Alassafi, and G. B. Wills, “Internet of Things Forensics: A Review,” Internet of Things, vol. 11, p. 100220, 2020.
- [141] H. F. Atlam, G. B. Wills, A. Alenezi, and M. O. Alassafi, “Blockchain with Internet of Things: Benefits, Challenges, and Future Directions,” Int. J. Intell. Syst. Appl., 2018.
- [142] C. Wang, S. Chen, Z. Feng, Y. Jiang, and X. Xue, “Block chain-based data audit and access control mechanism in service collaboration,” in Proceedings - 2019 IEEE International Conference on Web Services, ICWS 2019 - Part of the 2019 IEEE World Congress on Services, 2019, pp. 214–218.
- [143] H. Halpin and M. Piekarska, “Introduction to security and privacy on the blockchain,”

- in Proceedings - 2nd IEEE European Symposium on Security and Privacy Workshops, EuroS and PW 2017, 2017.
- [144] M. Chowdhury, S. Ferdous, and K. Biswas, "Blockchain Platforms for IoT Use-cases," no. July, pp. 3–4, 2018.
- [145] H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel, "Survey of platforms for massive IoT," in 2018 IEEE International Conference on Future IoT Technologies, Future IoT 2018, 2018.
- [146] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," IEEE Internet Things J., 2017.
- [147] M. Leo, F. Battisti, M. Carli, and A. Neri, "A federated architecture approach for Internet of Things security," in 2014 Euro Med Telco Conference - From Network Infrastructures to Network Fabric: Revolution at the Edges, EMTC 2014, 2014.
- [148] J. Decuir, "Introducing bluetooth smart: Part 1: A look at both classic and new technologies," IEEE Consumer Electronics Magazine. 2014.
- [149] F. Samie, L. Bauer, and J. Henkel, "IoT technologies for embedded computing: A survey," in 2016 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2016, 2016.
- [150] B. Kang, D. Kim, and H. Choo, "Internet of Everything: A Large-Scale Autonomic IoT Gateway," IEEE Trans. Multi-Scale Comput. Syst., vol. 3, no. 3, pp. 206–214, 2017.
- [151] K. Biswas and V. Muthukkumarasamy, "Securing smart cities using blockchain technology," in Proceedings - 18th IEEE International Conference on High Performance Computing and Communications, 14th IEEE International Conference on Smart City and 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, 2017.
- [152] R. Bryant, R. Katz, and E. Lazowska, "Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society," Comput. Res. Assoc., 2008.
- [153] I. K. Azeemi, M. Lewis, and T. Tryfonas, "Migrating to the cloud: Lessons and limitations of 'traditional' success models," in Procedia Computer Science, 2013, vol. 16, pp. 737–746.
- [154] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," in Proceedings - 10th International Conference on Frontiers of Information Technology, FIT 2012, 2012.
- [155] Z. Yang, Y. Yue, Y. Yang, Y. Peng, X. Wang, and W. Liu, "Study and application on the architecture and key technologies for IOT," in 2011 International Conference on Multimedia Technology, ICMT 2011, 2011.
- [156] M. Crosby, Nachiappan, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain Technology - BEYOND BITCOIN," Berkley Eng., 2016.
- [157] S. Davidson, P. De Filippi, and J. Potts, "Economics of Blockchain," SSRN Electron. J., 2016.
- [158] C. Khan, A. Lewis, E. Rutland, C. Wan, K. Rutter, and C. Thompson, "A Distributed-Ledger Consortium Model for Collaborative Innovation," Computer (Long. Beach. Calif.), 2017.

- [159] M. Benchoufi, R. Porcher, and P. Ravaud, "Blockchain protocols in clinical trials: Transparency and traceability of consent," F1000Research, 2018.
- [160] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in Proceedings - 2016 2nd International Conference on Open and Big Data, OBD 2016, 2016.
- [161] M. Wazid, A. K. Das, R. Hussain, G. Succi, and J. J. P. C. Rodrigues, "Authentication in cloud-driven IoT-based big data environment: Survey and outlook," J. Syst. Archit., 2019.
- [162] A. Mhenni, E. Cherrier, C. Rosenberger, and N. Essoukri Ben Amara, "Double serial adaptation mechanism for keystroke dynamics authentication based on a single password," Comput. Secur., 2019.
- [163] S. C. Cha, J. F. Chen, C. Su, and K. H. Yeh, "A Blockchain Connected Gateway for BLE-Based Devices in the Internet of Things," IEEE Access, 2018.
- [164] T. Sanda and H. Inaba, "Proposal of new authentication method in Wi-Fi access using Bitcoin 2.0," in 2016 IEEE 5th Global Conference on Consumer Electronics, GCCE 2016, 2016.
- [165] V. Venkatesh, J. Y. L. Thong, and X. Xu, "A I S ssoication for nformation ystems Unified Theory of Acceptance and Use of Technology: A Synthesis and the Road Ahead," J ournal, 2016.
- [166] A. H. Mohsin et al., "Blockchain authentication of network applications: Taxonomy, classification, capabilities, open challenges, motivations, recommendations and future directions," Computer Standards and Interfaces. 2019.
- [167] P. Kianmajd, J. Rowe, and K. Levitt, "Privacy-preserving coordination for smart communities," in Proceedings - IEEE INFOCOM, 2016.
- [168] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015, 2015.
- [169] K. Peterson, R. Deeduvanu, P. Kanjamala, and K. Boles, "A Blockchain-Based Approach to Health Information Exchange Networks," Proc. NIST Work. Blockchain Healthc., 2016.
- [170] S. Moin, A. Karim, Z. Safdar, K. Safdar, E. Ahmed, and M. Imran, "Securing IoTs in distributed blockchain: Analysis, requirements and open issues," Futur. Gener. Comput. Syst., 2019.
- [171] K. Wüst and A. Gervais, "Do you need a Blockchain?," IACR Cryptol. ePrint Arch., 2017.
- [172] S. Apte and N. Petrovsky, "Will blockchain technology revolutionize excipient supply chain management?," Journal of Excipients and Food Chemicals, 2016.
- [173] M. Banerjee, J. Lee, and K. K. R. Choo, "A blockchain future for internet of things security: a position paper," Digit. Commun. Networks, 2018.
- [174] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain Based Data Integrity Service Framework for IoT Data," in Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017, 2017.
- [175] K. Scarfone and M. Tracy, "Guide to General Server Security," Natl. Inst. Stand.

- Technol., 2008.
- [176] F. Loukil, M. Abed, and K. Boukadi, “Blockchain adoption in education: a systematic literature review,” *Educ. Inf. Technol.*, vol. 26, no. 5, pp. 5779–5797, 2021.
 - [177] L. Ao, C. Ogah, P. Asuquo, H. Cruickshank, and S. Zhili, “A Secure Key Management Scheme for Heterogeneous A Secure Key Management Scheme for Heterogeneous Secure Vehicular Communication Systems Secure Vehicular Communication Systems,” *ZTE Commun.*, vol. 21, no. 3, 2016.
 - [178] S. S. Panda, D. Jena, B. K. Mohanta, S. Ramasubbareddy, M. Daneshmand, and A. H. Gandomi, “Authentication and Key Management in Distributed IoT Using Blockchain Technology,” *IEEE Internet Things J.*, vol. 8, no. 16, 2021.
 - [179] O. Pal, B. Alam, V. Thakur, and S. Singh, “Key management for blockchain technology,” *ICT Express*, vol. 7, no. 1, 2021.
 - [180] S. Tavonatti, D. Battulga, M. Farhadi, C. Caprini, and D. Miorandi, “An experimental evaluation of the scalability of permissioned blockchains,” in *Proceedings - 2021 International Conference on Future Internet of Things and Cloud, FiCloud 2021*, 2021.
 - [181] M. H. Nasir, J. Arshad, M. M. Khan, M. Fatima, K. Salah, and R. Jayaraman, “Scalable blockchains — A systematic review,” *Futur. Gener. Comput. Syst.*, vol. 126, 2022.
 - [182] J. Bruce, “Purely P2P Crypto-Currency With Finite Mini-Blockchain,” *Bitfreak.Info*, no. May, 2013.
 - [183] A. Marsalek, T. Zefferer, E. Fasllija, and D. Ziegler, “Tackling data inefficiency: Compressing the bitcoin blockchain,” in *Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019*, 2019.
 - [184] Z. Wang et al., “A Data Lightweight Scheme for Parallel Proof of Vote Consensus,” *Proc. - 2021 IEEE Int. Conf. Big Data, Big Data 2021*, pp. 3656–3662, 2021.
 - [185] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, “OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding,” in *Proceedings - IEEE Symposium on Security and Privacy*, 2018, vol. 2018-May.
 - [186] H. Dang, T. T. A. Dinh, D. Loghin, E. C. Chang, Q. Lin, and B. C. Ooi, “Towards scaling blockchain systems via sharding,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2019.
 - [187] J. Hellings and M. Sadoghi, “Byshard: Sharding in a byzantine environment,” in *Proceedings of the VLDB Endowment*, 2021, vol. 14, no. 11.
 - [188] H. Huang et al., “Elastic Resource Allocation Against Imbalanced Transaction Assignments in Sharding-Based Permissioned Blockchains,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2372–2385, 2022.
 - [189] E. Madill, B. Nguyen, C. K. Leung, and S. Rouhani, *ScaleSFL: A Sharding Solution for Blockchain-Based Federated Learning*, vol. 1, no. 1. Association for Computing Machinery, 2022.
 - [190] L. Aumayr, K. Abbaszadeh, and M. Maffei, “Thora : Atomic And Privacy-Preserving Multi-Channel Updates,” *IACR*, pp. 1–25, 2022.

- [191] S. Dangi, A. Aggarwal, and P. Rastogi, "Integrating Blockchain with Education: Proposed Model, Prospects and Challenges," in *Transformations Through Blockchain Technology*, 2022.
- [192] Institute of Electrical and Electronics Engineers (IEEE), "ISO/IEC/IEEE International Standard - Systems and software engineering - Vocabulary [PDF file]," Iso/Iec/Ieee 24765, 2017.
- [193] F. Peng and X. Jiang, "A novel education system requirements engineering methodology," in *ITME 2011 - Proceedings: 2011 IEEE International Symposium on IT in Medicine and Education*, 2011.
- [194] M. B. Ila and H. Kitapci, "Selecting an effective information and communication technology architecture for an education system based on non-functional requirements," in *8th IEEE International Conference on Application of Information and Communication Technologies, AICT 2014 - Conference Proceedings*, 2014.
- [195] H. M. Hussien, S. M. Yasin, S. N. I. Udzir, A. A. Zaidan, and B. B. Zaidan, "A Systematic Review for Enabling of Develop a Blockchain Technology in Healthcare Application: Taxonomy, Substantially Analysis, Motivations, Challenges, Recommendations and Future Direction," *J. Med. Syst.*, 2019.
- [196] N. Satoshi and S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic cash system," *Bitcoin*, 2008.
- [197] M. Han, D. Wu, Z. Li, Y. Xie, J. S. He, and A. Baba, "A novel blockchain-based education records verification solution," in *SIGITE 2018 - Proceedings of the 19th Annual SIG Conference on Information Technology Education*, 2018.
- [198] H. Dai et al., "TrialChain: A blockchain-based platform to validate data integrity in large, biomedical research studies," *arXiv*. 2018.
- [199] J. P. Miguel, D. Mauricio, and G. Rodríguez, "A Review of Software Quality Models for the Evaluation of Software Products," *Int. J. Softw. Eng. Appl.*, 2014.
- [200] B. Mackenzie, R. I. Ferguson, and X. Bellekens, "An Assessment of Blockchain Consensus Protocols for the Internet of Things," in *2018 International Conference on Internet of Things, Embedded Systems and Communications, IINTEC 2018 - Proceedings*, 2018.
- [201] F. B. Vernadat, "Interoperable enterprise systems: Architectures and methods," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2006.
- [202] E. Abebe et al., "Enabling Enterprise Blockchain Interoperability with Trusted Data Transfer (industry track)," in *Middleware Industry 2019 - Proceedings of the 2019 20th International Middleware Conference Industrial Track, Part of Middleware 2019*, 2019.
- [203] S. Khan, M. B. Amin, A. T. Azar, and S. Aslam, "Towards Interoperable Blockchains: A Survey on the Role of Smart Contracts in Blockchain Interoperability," *IEEE Access*, 2021.
- [204] H. Y. Paik, X. Xu, H. M. N. D. Bandara, S. U. Lee, and S. K. Lo, "Analysis of data management in blockchain-based systems: From architecture to governance," *IEEE Access*, 2019.
- [205] A. Shanker, "Public vs. private blockchains," *PC Magazine*, 2017.
- [206] J. Schneider et al., "Blockchain - Putting Theory into Practice," 2016.

- [207] J. King, “ERIS,” in ERIS, 2020.
- [208] M. Hearn, “Corda: A distributed ledger,” Whitepaper, 2016.
- [209] C. Pahl, N. El Ioini, and S. Helmer, “A decision framework for blockchain platforms for iot and edge computing,” in IoTBDS 2018 - Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security, 2018.
- [210] S. Jeong, “Dogecoin,” in Paid: Tales of Dongles, Checks, and Other Money Stuff, 2017.
- [211] Litecoin.info, “Litecoin,” litecoin.info, 2018. .
- [212] S. King, “Primecoin: Cryptocurrency with Prime Number Proof-of-Work,” King, Sunny, 2013.
- [213] D. Stone, “Delayed blockchain protocols,” arXiv. 2018.
- [214] QuantumMechanic, “Proof of Stake Instead of Proof of Work,” GitHub, 2011.
- [215] F. Yang, W. Zhou, Q. Wu, R. Long, N. N. Xiong, and M. Zhou, “Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism,” IEEE Access, 2019.
- [216] L. Ren, “Proof of Stake Velocity: Building the Social Currency of the Digital Age,” 2014.
- [217] K. Karantias, A. Kiayias, and D. Zindros, “Proof-of-Burn,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2020.
- [218] Hyperledger sawtooth, “Hyperledger Sawtooth,” <https://Sawtooth.Hyperledger.Org>. 2019.
- [219] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” ACM Trans. Program. Lang. Syst., 1982.
- [220] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” Proc. Symp. Oper. Syst. Des. Implement., 1999.
- [221] L. Ismail and H. Materwala, “A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions,” Symmetry (Basel)., vol. 11, no. 10, 2019.
- [222] M. Vukolić, “The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016.
- [223] J. R. Douceur, “The sybil attack,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2002.
- [224] Q. Wang et al., “Security Analysis on dBFT Protocol of NEO,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2020.
- [225] D. Mazieres and D. Mazières, “The stellar consensus protocol: A federated model for internet-level consensus,” Stellar Dev. Found., 2015.
- [226] S. Pahlajani, A. Kshirsagar, and V. Pachghare, “Survey on Private Blockchain Consensus Algorithms,” in Proceedings of 1st International Conference on Innovations

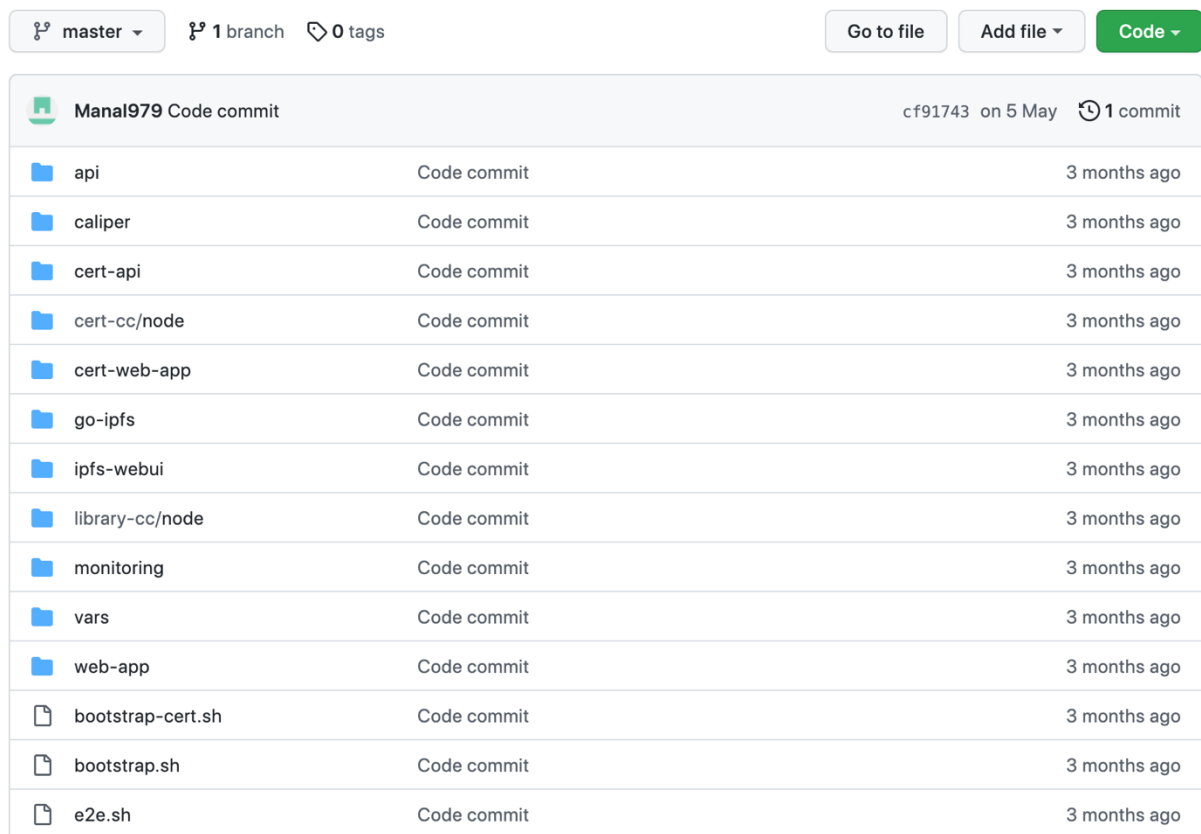
- in Information and Communication Technology, ICICT 2019, 2019.
- [227] JP Morgan Chase, “Quorum Whitepaper,” New York JP Morgan Chase, 2016.
- [228] R. G. Brown, “The Corda Platform: An Introduction,” Corda Platf. White Pap., 2018.
- [229] E. Androulaki et al., “Hyperledger fabric,” 2018.
- [230] Z. Li, R. Y. Zhong, Z. G. Tian, H. N. Dai, A. V. Barenji, and G. Q. Huang, “Industrial Blockchain: A state-of-the-art Survey,” *Robotics and Computer-Integrated Manufacturing*, vol. 70. 2021.
- [231] M. El Ghamry, I. T. A. Halim, and A. M. Bahaa-Eldin, “Secular: A Decentralized Blockchain-based Data Privacy-preserving Model Training Platform,” in 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference, MIUCC 2021, 2021.
- [232] A. I. Sanka, M. Irfan, I. Huang, and R. C. C. Cheung, “A survey of breakthrough in blockchain technology: Adoptions, applications, challenges and future research,” *Computer Communications*, vol. 169. 2021.
- [233] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, “Decentralized Applications: The Blockchain-Empowered Software System,” *IEEE Access*, 2018.
- [234] Stack Overflow Ltd., “Stack Overflow Developer Survey 2019,” Stack Overflow Insights, 2020. .
- [235] E. Androulaki et al., “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains,” in *Proceedings of the 13th EuroSys Conference, EuroSys 2018*, 2018.
- [236] V. Buterin et al., “A next-generation smart contract and decentralized application platform,” *PLoS One*, 2018.
- [237] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, “Survey of consensus protocols on blockchain applications,” in 2017 4th International Conference on Advanced Computing and Communication Systems, ICACCS 2017, 2017.
- [238] S. Y. Lim et al., “Blockchain Technology the Identity Management and Authentication Service Disruptor: A Survey,” *Int. J. Adv. Sci. Eng. Inf. Technol.*, 2018.
- [239] A. Badr, L. Rafferty, Q. H. Mahmoud, K. Elgazzar, and P. C. K. Hung, “A permissioned blockchain-based system for verification of academic records,” in 2019 10th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2019 - Proceedings and Workshop, 2019.
- [240] T. Hepp, M. Sharinghousen, P. Ehret, A. Schoenhals, and B. Gipp, “On-chain vs. off-chain storage for supply- and blockchain integration,” *it - Inf. Technol.*, 2018.
- [241] I. Baumgart and S. Mies, “IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3),” *Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS*, 2007.
- [242] R. A. Mishra, A. Kalla, A. Braeken, and M. Liyanage, “Privacy Protected Blockchain Based Architecture and Implementation for Sharing of Students’ Credentials,” *Inf. Process. Manag.*, vol. 58, no. 3, 2021.
- [243] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, “Performance analysis of private blockchain platforms in varying workloads,” in 2017 26th International Conference on Computer Communications and Networks, ICCCN 2017, 2017.

- [244] J. Jeong, D. Kim, S. Y. Ihm, Y. Lee, and Y. Son, "Multilateral Personal Portfolio Authentication System Based on Hyperledger Fabric," *ACM Trans. Internet Technol.*, vol. 21, no. 1, 2021.
- [245] E. Mansour, F. Shahzad, J. Tekli, and R. Chbeir, "Data Redundancy Management in Connected Environments," in *Q2SWinet 2020 - Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2020.
- [246] M. Sirajudeen Yoosuf, R. Anitha, and M. S. Yoosuf, "Low Latency Fog-Centric Deduplication Approach to Reduce IoT Healthcare Data Redundancy," *Wirel. Pers. Commun.*, pp. 1–23, 2022.

Appendix

The source code for the implementations and the testing can be found at GitHub <http://github.com/Manal979/Project>.

The Codebase:



master 1 branch 0 tags

Go to file Add file Code

Manal979 Code commit		cf91743 on 5 May 1 commit
api	Code commit	3 months ago
caliper	Code commit	3 months ago
cert-api	Code commit	3 months ago
cert-cc/node	Code commit	3 months ago
cert-web-app	Code commit	3 months ago
go-ipfs	Code commit	3 months ago
ipfs-webui	Code commit	3 months ago
library-cc/node	Code commit	3 months ago
monitoring	Code commit	3 months ago
vars	Code commit	3 months ago
web-app	Code commit	3 months ago
bootstrap-cert.sh	Code commit	3 months ago
bootstrap.sh	Code commit	3 months ago
e2e.sh	Code commit	3 months ago