

**ROBOTIC PROCESS AUTOMATION IN SECURE SUPPLY
CHAIN MANAGEMENT**
BY
CHEANG KIT YANN

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF INFORMATION SYSTEMS (HONOURS) DIGITAL ECONOMY
TECHNOLOGY
Faculty of Information and Communication Technology
(Kampar Campus)

JUNE 2024

REPORT STATUS DECLARATION FORM

Title: Robotic Process Automation in Secure Supply Chain Management

Academic Session: June 2024

I CHEANG KIT YANN

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

42, Jalan Mas ½,
Taman Mas 1, Kampung Koh,
32000, Sitiawan,
Perak

Ts Dr. Ooi Joo On
Supervisor's name

Date: 27th August 2024

Date: 11/09/2024

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN**

Date: 27th August 2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Cheang Kit Yann (ID No: 20ACB03597) has completed this final year project entitled “*Robotic Process Automation in Secure Supply Chain Management*” under the supervision of Ts Dr. Ooi Joo On (Supervisor) from the Department of Digital Economy, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,




Cheang Kit Yann

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled “**ROBOTIC PROCESS AUTOMATION IN SECURE SUPPLY CHAIN MANAGEMENT**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : Cheang Kit Yann

Date : 27th August 2024

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Ts Dr. Ooi Joo On and Ts Deveendra Menon a/l Narayanan Nair who has given me this bright opportunity to engage in an IC design project. It is my first step to establish a career in IC design field. Besides that, they have given me a lot of guidance in order to complete this project. When I was facing problems in this project, the advice from them always assists me in overcoming the problems. Again, a million thanks to my supervisor and moderator.

Other than that, to a very special person in my life, Lim Ming Xuan, for his patience, unconditional support, and love, and for standing by my side during hard times, and provided a lot of support to me when completing this project. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

This project is a development-based project with automation for academic purposes. The integration aims to enable organizations to automate security assessments and enhance overall supply chain security. Given that RPA is a relatively new technology, this thesis includes a systematic literature review of similar platforms and technologies available in the market. Current development challenges include financial constraints, security risks, privacy concerns, and complexities in integrating RPA with SCM. Various blockchain platforms and smart contracts are reviewed, with a focus on designing and integrating blockchain technology into SCM processes. The reviewed blockchain platforms include Ethereum, Hyperledger Fabric, Quorum, and Corda. Besides, it explores the application of blockchain in SCM, emphasizing its potential to enhance transparency and efficiency. Furthermore, it reviews several companies IBM Food Trust, Provenance, Walmart, and TradeWaltz that have successfully incorporated blockchain technology into their supply chain operations, showcasing real-world benefits and implementations. This project details the system design, focusing on Hyperledger Fabric's transaction workflow and the supply chain architecture, along with the Hyperledger Firefly development stack. Last of all, the expected outcome of this project is to integrate RPA and blockchain technology into supply chain management to improve security and overall efficiency.

Keywords: Robotic Process Automation, Supply Chain Security, Blockchain Integration, Hyperledger Firefly, Smart Contract Technology, Automation in SCM, Blockchain Platforms, Security Assessments, SCM Transparency, Blockchain Implementation, Hyperledger Fabric Workflow, SCM Efficiency, Real-World Blockchain Applications, Blockchain for Security, RPA in Supply Chain Management

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	x
LIST OF TABLES	xiv
LIST OF ABBREVIATIONS	xv
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	5
1.2 Objectives	6
1.3 Project Scope and Direction	7
1.4 Contributions	7
1.5 Report Organization	8
CHAPTER 2 LITERATURE REVIEW	10
2.1 Background to Smart Contracts and Blockchain Platforms	10
2.1.1 Ethereum	13
2.1.2 Hyperledger Fabric	15
2.1.3 Quorum	18
2.1.4 Corda	20
2.2 Summary of Reviewed Blockchain Technology	22
2.3 Supply Chain Management using Blockchain Technology	24

2.4	Use case of Blockchain Based SCM	27
2.4.1	IBM Food Trust	27
2.4.2	Provenance	34
2.4.3	Walmart	36
2.4.4	TradeWaltz	38
CHAPTER 3 SYSTEM METHODOLOGY/APPROACH		40
3.1	System Requirement	41
3.1.1	Hardware	41
3.1.2	Software/Tools	42
CHAPTER 4 SYSTEM DESIGN		45
4.1	System Design Diagram	45
4.1.1	Overview of Hyperledger Fabric	45
4.2	Transaction Workflow of Hyperledger Fabric	48
4.2.1	Execute	49
4.2.2	Order	49
4.2.3	Validate	50
4.3	Supply Chain Architecture Diagram	51
4.4	Hyperledger Firefly Development Stack	52
CHAPTER 5 SYSTEM IMPLEMENTATION		53
5.1	Setting up	53
5.1.1	Software	53
5.2	Implementation on Kaleido Platform	54
5.3	Implementation Using Ubuntu Linux Virtual Machine	85
5.3.1	Generating Network and Invoking Smart Contracts	85
5.3.2	Initializing Hyperledger Firefly CLI	93
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION		104
6.1	System Testing and Performance Metrics	104
6.2	Testing Setup and Result	108

6.3	Project Challenges	109
6.4	Objectives Evaluation	110
6.5	Concluding Remark	111
CHAPTER 7 CONCLUSION AND RECOMMENDATION		112
7.1	Conclusion	112
7.2	Recommendations	113
REFERENCES		113
WEEKLY LOG		122
POSTER		126
PLAGIARISM CHECK RESULT		127
FYP2 CHECKLIST		133

LIST OF FIGURES

Figure Number	Title	Page
Figure 1.0.1	Centralized Supply Chain Inventory System	3
Figure 1.0.2	Decentralized Supply Chain Inventory System	3
Figure 1.0.3	Distributed Ledger Technology	4
Figure 1.0.4	Structure of DLT	4
Figure 2.1.1	Structure of Smart Contract	10
Figure 2.1.2	How RPA Enhances Smart Contract Efficiency	11
Figure 2.1.3	Smart Contract Execution	12
Figure 2.1.1.1	Ethereum	13
Figure 2.1.2.1	Hyperledger Fabric	15
Figure 2.1.2.2	Blockchain Technology for Smart Contracts	16
Figure 2.1.2.3	Execute-order-validate architecture of Fabric	17
Figure 2.1.3.1	Quorum	18
Figure 2.1.4.1	Corda	20
Figure 2.3.1	Blockchain-based Supply Chain Management	24
Figure 2.3.2	Blockchain Utilization in Supply Chain	25
Figure 2.4.1.1	Blockchain Role in Supply Chain	28
Figure 2.4.1.2	IBM Food Trust Solution	29
Figure 2.4.1.3	IBM Food Trust Trace Module (a)	29
Figure 2.4.1.4	IBM Food Trust Trace Module (b)	30
Figure 2.4.1.5	IBM Food Trust Fresh Insights Module	31
Figure 2.4.1.6	IBM Food Trust Certification Module (a)	32
Figure 2.4.1.7	IBM Food Trust Certification Module (b)	32
Figure 2.4.2.1	Provenance of Tuna	35
Figure 2.4.4.1	TradeWaltz Trade Ecosystem	39
Figure 3.0	SSDLC Process	40
Figure 4.1.1.1	A Fabric network with federated MSPs and running multiple (differently shaded and coloured) chaincodes	45
Figure 4.1.1.2	Components of Fabric Peer	46

Figure 4.2.1	Transaction Workflow of Fabric	48
Figure 4.3.1	Managing Supply Chain Process using Blockchain Technology	51
Figure 4.4.1	Firefly Development Stack	52
Figure 5.1.1.1	Sign Up	53
Figure 5.2.1	Main Page of Kaleido	54
Figure 5.2.2	Create Network	54
Figure 5.2.3	Home Region	55
Figure 5.2.4	Kaleido Dashboard	55
Figure 5.2.5	Environment Type	56
Figure 5.2.6	Environment Protocol	56
Figure 5.2.7	Environment Details	57
Figure 5.2.8	Environment Provider Settings	57
Figure 5.2.9	Environment Dashboard (a)	58
Figure 5.2.10	Creation of Node	58
Figure 5.2.11	Setup Certificate Authority (CA)	59
Figure 5.2.12	Node Configuration	59
Figure 5.2.13	Environment Dashboard (b)	59
Figure 5.2.14	Environment Dashboard (c)	60
Figure 5.2.15	Create IPFS	60
Figure 5.2.16	IPFS Dashboard	61
Figure 5.2.17	Channel Page	61
Figure 5.2.18	Create Channel	62
Figure 5.2.19	Add Channel Members	62
Figure 5.2.20	Apps Page (a)	63
Figure 5.2.21	Create App Page	63
Figure 5.2.22	Enter App Details	64
Figure 5.2.23	App Page (b) - Asset Transfer Created	64
Figure 5.2.24	Enter Chaincode Details	65
Figure 5.2.25	App: Asset Transfer Version Details	66
Figure 5.2.26	Promote to Environment	66
Figure 5.2.27	Deploy Instance to Channel	67
Figure 5.2.28	Select Channel	67

Figure 5.2.29	Chaincode Page	68
Figure 5.2.30	Blockchain Dashboard	68
Figure 5.2.31	Peer1 Logs	69
Figure 5.2.32	Peer Node Overview	69
Figure 5.2.33	Fabric Blockchain Explorer - Sign in Page	70
Figure 5.2.34	Fabric Blockchain Explorer Webpage Dashboard	70
Figure 5.2.35	Fabric Blockchain Explorer - Channels Page	71
Figure 5.2.36	Node Page – View Rest API	71
Figure 5.2.37	Rest API Gateway	72
Figure 5.2.38	POST /identities (a)	72
Figure 5.2.39	POST /identities (b)	73
Figure 5.2.40	POST /identities (c) – Server Response	73
Figure 5.2.41	POST /identities/{username}/enroll (a)	74
Figure 5.2.42	POST /identities/{username}/enroll (b) – Server Response	75
Figure 5.2.43	GET /identities/{username} (a)	75
Figure 5.2.44	GET /identities/{username} (b)	76
Figure 5.2.45	POST /transactions (a)	78
Figure 5.2.46	POST /transactions (b) – Server Response	78
Figure 5.2.47	POST /transactions (c) – Async Mode	79
Figure 5.2.48	POST /transactions (d) – Async Mode Server Response	80
Figure 5.2.49	GET /receipts/{receiptId} (a) - Async Mode	80
Figure 5.2.50	GET /receipts/{receiptId} (b) – Async Mode Server Response	81
Figure 5.2.51	POST /eventstreams (a)	81
Figure 5.2.52	POST /eventstreams (b) – Server Response	82
Figure 5.2.53	POST /subscriptions (a)	83
Figure 5.2.54	POST /subscriptions (b) – Server Response	84
Figure 5.3.1.1	Creating Channel using network.sh script	85
Figure 5.3.1.2	Channel Successfully Joined	86
Figure 5.3.1.3	Deploying Chaincode to Instances (a)	86
Figure 5.3.1.4	Deploying Chaincode to Instances (b)	87
Figure 5.3.1.5	Initializing Ledger with Assets	88

Figure 5.3.1.6	Query the Ledger on Peer0.Org1	89
Figure 5.3.1.7	Invoking Chaincode	90
Figure 5.3.1.8	Query the Ledger on Peer0.Org2	90
Figure 5.3.1.9	Bring up CAs for Organizations in the Network	91
Figure 5.3.1.10	MSP Folder of Org1 Admin User	92
Figure 5.3.2.1	Creating Firefly Development Stack	93
Figure 5.3.2.2	Starting Firefly Stack	94
Figure 5.3.2.3	Web UI Interface – Firefly Explorer (a)	95
Figure 5.3.2.4	Web UI Interface – Firefly Explorer (b)	95
Figure 5.3.2.5	Firefly Sandbox (a)	96
Figure 5.3.2.6	Firefly Sandbox (b)	96
Figure 5.3.2.7	Swagger API UI (a)	97
Figure 5.3.2.8	Swagger API UI (b)	97
Figure 5.3.2.9	Member 0 Sending Broadcast Message	98
Figure 5.3.2.10	Broadcast Message Received	99
Figure 5.3.2.11	Member 1 Firefly Explorer – Broadcast Message	99
Figure 5.3.2.12	Member 1 Sending Private Message	100
Figure 5.3.2.13	Member 0 Firefly Explorer – Private Message	101
Figure 5.3.2.14	Member 2 Firefly Explorer – Private Message	101
Figure 5.3.2.15	Member 3 Firefly Explorer – Private Message	101
Figure 5.3.2.16	Member 0 Sending Private File	103
Figure 5.3.2.17	Member 1 – Download the .txt File	103
Figure 6.1.1	VM - Ubuntu Version 20.04.6	105
Figure 6.1.2	Kaleido Platform	106
Figure 6.1.3	WSL	106
Figure 6.1.4	Docker Version	107
Figure 6.1.5	Docker Compose Version	107
Figure 6.1.6	Go Version	107
Figure 6.1.7	JavaScript Version	107
Figure 6.1.8	Node.js Version	107

LIST OF TABLES

Table Number	Title	Page
Table 2.1.1.1	Summary of Ethereum Authentication and Authorization Schemes	13
Table 2.1.2.1	Summary of Hyperledger Fabric Authentication and Authorization Schemes	15
Table 2.1.3.1	Summary of Quorum Authentication and Authorization Schemes	18
Table 2.1.4.1	Summary of Corda Authentication and Authorization Schemes	20

LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>API</i>	Application Programming Interface
<i>BaaS</i>	Blockchain as a Service
<i>BC</i>	Blockchain
<i>CA</i>	Certificate Authorities
<i>CBDC</i>	Central Bank Digital Currency
<i>DLT</i>	Distributed Ledger Technology
<i>ECDSA</i>	Elliptic Curve Digital Signature Algorithm
<i>EVM</i>	Ethereum Virtual Machine
<i>IoT</i>	Internet of Things
<i>IPFS</i>	InterPlanetary File System
<i>IPNLF</i>	International Pole and Line Foundation
<i>JVM</i>	Java Virtual Machine
<i>ML</i>	Machine Learning
<i>MSP</i>	Membership Service Provider
<i>OS</i>	Operating System
<i>pBFT</i>	Practical of Byzantine Fault Tolerance
<i>PoS</i>	Proof of Stake
<i>PoW</i>	Proof of Work
<i>RPA</i>	Robotic Process Automation
<i>SaaS</i>	Software as a Service
<i>SC</i>	Supply Chain
<i>SCM</i>	Supply Chain Management
<i>SCS</i>	Supply Chain Security
<i>SDK</i>	Software Development Kit
<i>SSDLC</i>	Secure Software Development Lifecycle
<i>TSA</i>	Transaction Sender Authentication
<i>UI</i>	User Interface
<i>VM</i>	Virtual Machine
<i>WSL</i>	Windows Subsystem for Linux

Chapter 1

Introduction

This chapter presents an overview background towards RPA, SCM, security in the supply chain, blockchain technology, objectives, project scope and direction, followed by the contributions of this project towards secure SCM and last of all, the report organizations.

Robotic Process Automation

The emergence of a variety of digital technologies has been evolving at a fast pace in a way that the rise of digital transformation is an unavoidable trend occurring all around the world. It provides countless opportunities and products that contribute to the modernization era [1]. RPA is one of the significant outcomes as a result of the advanced technology. The word R stands for robotics, contributing to that robots would mimic activities done by humans based on predefined rules, P stands for process, whereby the robots are running a series of tasks, and A stands for automation, where the demand of humans in carrying out the processes can be eliminated and replaced by robots [6]. The bloom of RPA has greatly impacted the way organizations, companies, and industries operate their businesses from the traditional way to step into the journey of digital business transformation [2]. RPA is a software-based solution focusing on simplifying human tasks in the organization, aiming to streamline operations, automate repetitive tasks traditionally performed by humans, and enhance overall business efficiency by reducing human errors and staffing costs [3] – [5].

Supply Chain Management

Meanwhile, RPA has gradually gained attention as experts have embraced this disruptive technology for utilization in SCM security initiatives [1]. SCM is the process that encompasses all production flow of goods and services, inclusive of procurement of raw materials, and the process of transforming the raw materials to delivering end-products to consumers [8] - [10]. This involves coordinating and collaborating with channel partners, including suppliers, intermediaries, third-party service providers, and customers. On top of that, Supply Chain Security (SCS) is mainly to minimize the security risk of using software developed by other organizations while securing data

CHAPTER 1

access from other parties [12]. Experts state that the application of RPA in the SC will bring tremendous improvement in the supply chain ecosystems, especially the way in identifying, analyzing, and mitigating the risks and security related to products and cybersecurity [7], [12]. In accordance with the MHI Annual Industry Report (2019), among the top influence technologies, RPA is expected to have a profound impact on SCM, with an 87% prediction rate in the decade to come [1]. Automation processes are capable of enhancing security processes by substituting robots to replace humans in configuring various security aspects of the SCM system [11]. Henceforth, the implementation of RPA is indispensable to organizations in SCM to engage in the dynamic nature of digital transformation.

Blockchain Technology

The blockchain technology defines the role of providing traceability, security, immutability, and eliminates the need for third-party access, creating a decentralized network among all users. Once the data is recorded, it cannot be altered as another block would have the record of the correct history transaction. Apart from Distributed Ledger Technology (DLT), blockchain technology comprises smart contracts to run automatically by setting predefined rules and conditions [18]. The assistance of blockchain technology could promote greater security and traceability to the system. In addition, the blockchain technology can be implemented into the supply chain system. The application of a smart contract provides a powerful framework in improving efficiency and trust in the operations as it is decentralized. All the transaction data is updated in the blockchain ledger to acknowledge every party involved [21]. As this immutable ledger technology can effectively prevent the involvement of cyberattacks, it helps to minimize the fraud risks and disputes not to mention creating a more dependable system [20].

There are two types of blockchain categories, permission and permissionless. To differentiate both categories, blockchain is deemed permission when users are to attain permission from the administrator in order to access the network [27]. An example is Hyperledger Fabric which uses the consensus protocol in determining the state of a ledger. On the contrary, blockchain is deemed permissionless when anyone can join a network with just their permission and conduct transactions [27]. For instance, Bitcoin

and Ethereum. As illustrated in the below figure, shows the overview of a centralized SC inventory system on the left side (a), and illustrates the decentralized SC inventory system on the right side (b).

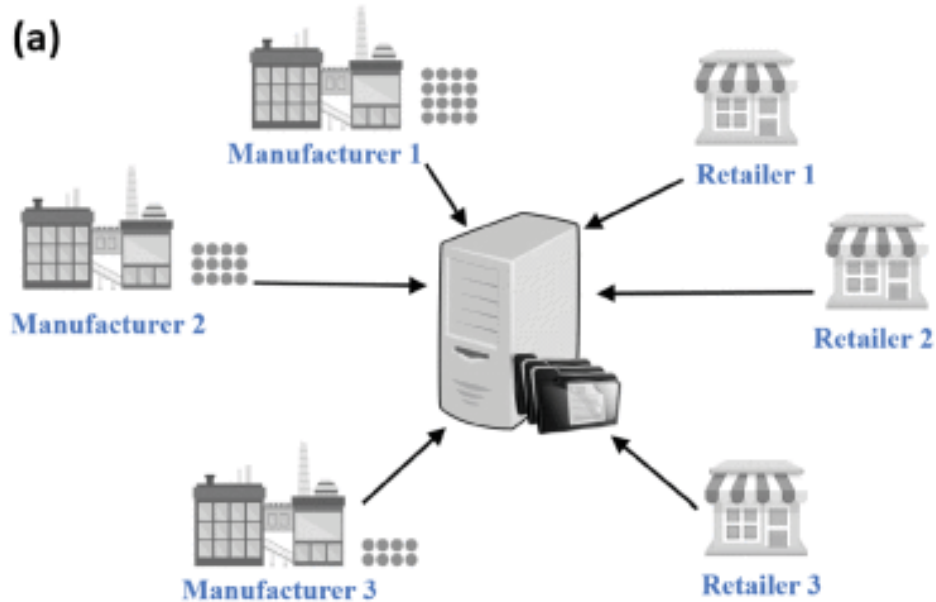


Figure 1.0.1 Centralized Supply Chain Inventory System

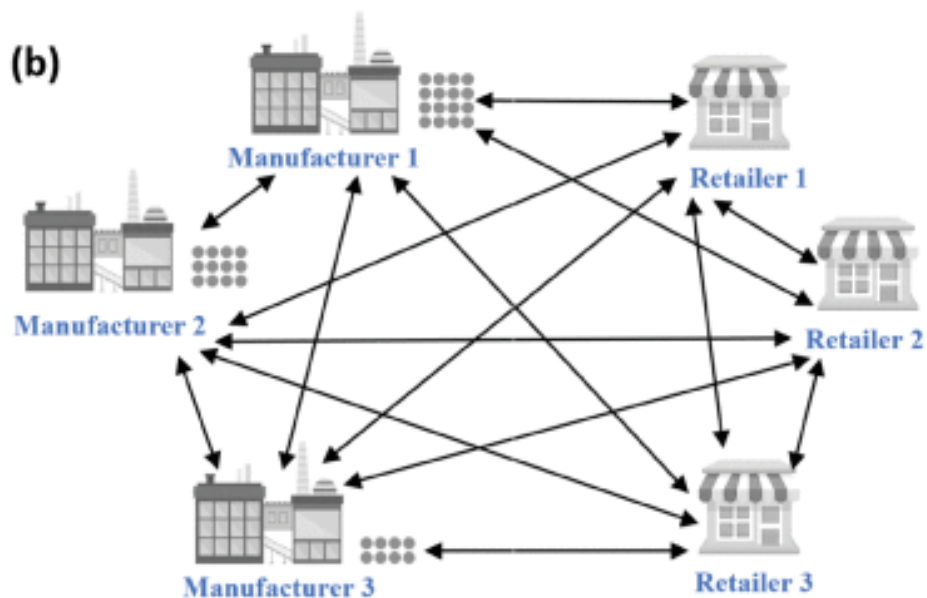


Figure 1.0.2 Decentralized Supply Chain Inventory System

[19] explained that blockchain is a technology that stores all the transaction records in different databases to create connectedness through peer-to-peer nodes. In other words,

CHAPTER 1

all the blocks are linked together and connected to the chain, which is known as the 'digital ledger' or 'distributed ledger technology (DLT)' as shown in Figure 1.0.2.

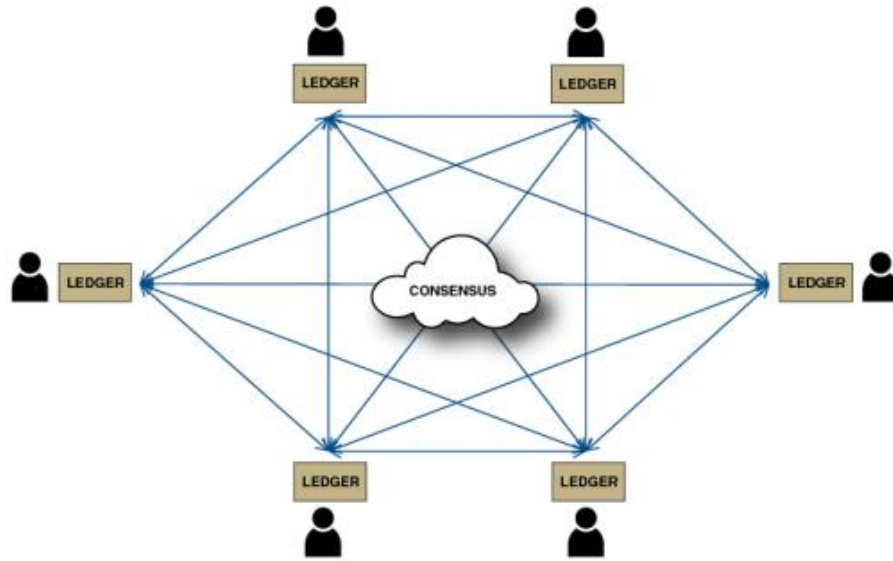


Figure 1.0.3 DLT

The digital ledger ensures the security of the transactions, by requesting the unique digital signature of the owner to authenticate the transaction and empowers every user to trace the history of transactions that have been made by themselves or others. Figure 1.0.3 explains the structure of the distributed ledger. According to the context of blockchain, states that each block is comprised of a timestamp, stored data, and a cryptographic hash of the one ahead of it, which links the blocks together to build a chain [17].

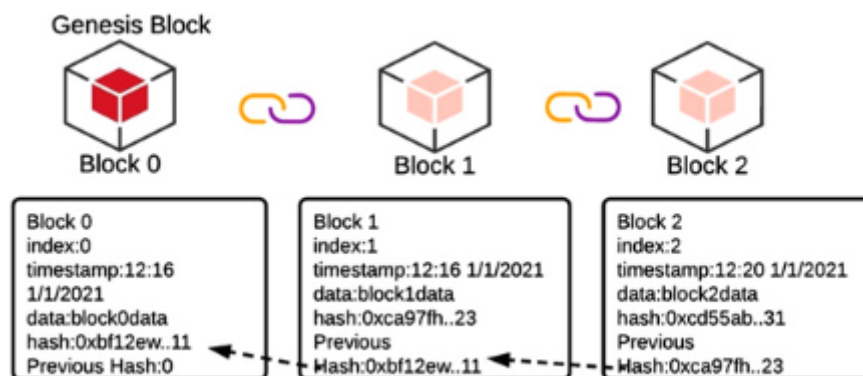


Figure 1.0.4 Structure of DLT

1.1 Problem Statement and Motivation

Challenges in selecting the right RPA software for effective implementation

Nowadays, organizations encountered challenges during the selection process of the appropriate RPA Software to maximize its best advantage for effective implementation. There are various RPA vendors in the market, offering a bundle of pricing models, and diversified features and functionality. Thereby, it is time-consuming and challenging to evaluate all the offered RPA solutions to cater for the needs of companies such as assessing the compatibility of RPA software with current systems, assessment of different business goals, scalability and preferences to name a few. It also leads to uncertainty to a certain extent that companies are having too many choices during the selection process, in the end hindering the efficient implementation of RPA within the organization.

Complexity of implementation to achieve SCM system security

Besides, the implementation of security measures within the SCM domain is inherently complex and laborious. It entails an excessive number of intricate and time-consuming steps in configuring the SCM system security, including access controls, data encryption, user authentication, and threat detection, among others. These steps not only place a substantial demand on time and resources, but also introduce a significant risk of human error, which could potentially affect the integrity and resilience of the SCM system's security. The inefficiency or error in these configurations could bring severe consequences to the entire security operations, and eventually decrease the efficiency.

Financial risks and barriers

Moving on next, the occurrence of financial risks and barriers is one of the main concerns of implementing RPA software in secure SCM. To ensure the functionality and longevity of the newly designed system, substantial financial resources are required to cover ongoing maintenance costs, investments in both software and hardware infrastructure, and continuous development to adapt to evolving technological advancements. These demands are invariably straining the budget, whether in long term

or short term. In addition, the manipulation of RPA software requires specialized skills and knowledge to apply in secure supply chain management, and thus, extra funds allocation is needed as well for employee training or hiring skilled employees, and the implementation of blockchain technology in the SCM.

Thus, the purpose of this study is to give a comprehensive overview and background of the proposed project, "RPA in Secure Supply Chain Management". The necessity to increase supply chain security and efficiency drove the development of this project. Due to the rapidly evolving era, the supply chain has increased its importance, and becoming more complex. This complexity has led to certain risks of data breaches, inefficiencies, vulnerabilities, and compliance issues. Thus, it inspires me to integrate RPA and blockchain technology into SCM to improve security and overall efficiency.

1.2 Objectives

To automate processes in the supply chain

One of the objectives is to automate processes in the supply chain. By replacing manual processes with automated workflows, organizations can reduce errors, enhance operational efficiency, and improve real-time monitoring. Automation also strengthens security by continuously detecting risks and anomalies, enabling faster responses to threats and disruptions. Ultimately, automating these processes enhances both the performance and safety of the supply chain.

To boost overall efficiency in monitoring the supply chain security

The proposed methodology is targeted to enhance the efficiency of monitoring the supply chain process from the perspective of security. In current supply chain security processes, most of them are undergoing manual oversight, which can be time-consuming and certain rate of errors. By implementing RPA technology, automation can be configured and assist humans in various monitoring tasks such as real-time data analytics, threat detection, penetration and vulnerability testing, and access control verification [12]. This integration not only bolsters the security and detects breaches in

SCM, but also maximizes the benefit of RPA technology in helping stakeholders with decision-making within the supply chain security domain.

To achieve scalability and adaptability in supply chain security

This project aims to deliver an RPA-driven solution in supply chain security system that seamlessly aligns with the dynamic needs of organizations in the adaptation to changing security requirements. By focusing on scalability and adaptability, RPA systems can be flexibly adjusted to accommodate the demand or emerging security requirements. This adaptability empowers organizations can precisely respond to arising security challenges and ensure they can address the growing demand and complexities in the area of supply chain.

1.3 Project Scope and Direction

The scope of this project encompasses the architecture, algorithm and design on improving the overall efficiency and scalability of the SCM. By harnessing the RPA and blockchain technology, it helps to secure data access from third parties, at the same time ensuring a seamless execution throughout the entire system operations [12]. Besides, it increases the overall speed, efficiency, and accuracy of the work by providing a precise and minimizes error throughout the operation process. The automated robots are able to execute various automated processes with predefined rules and adhere to established rules and guidelines. As a result, this project eases overall streamlined operations in supply chain, allowing the transformation of how SCM is managed and secured.

1.4 Contributions

This project is specifically developing and designing the system with the integration of Robotic Process Automation (RPA) in secure SCM. It holds a significant contribution to the field of SCM. Firstly, the report highlights the challenges faced by organizations in monitoring the SCM. This offers researchers a comprehensive understanding and deeper insights into current problems encountered in the existing supply chain issues and facilitates more effective problem-solving. Apart from this, this project delves into

CHAPTER 1

the integration of RPA technology, which is deemed valuable and observable to be investigated by the researchers on ways to implement this technology into the SC framework.

This report serves as a valuable reference to the researchers if they are interested in implementing RPA in their own projects, and it outlines the methodology to be used throughout the development process. Next, the project serves as a catalyst to innovate, and to prompt further technological advancements. By showcasing the potential of RPA technology, it could inspire more people to delve into it and induce the growth of RPA technology.

In addition, this report aims to compare the most appropriate tools and platforms in terms of the blockchain technology to be implemented into the SCM. It delves into the different types of platforms within the blockchain and reviews the platform from the perspective of smart contract algorithms, categories of blockchain, consensus mechanisms, distinct ways of authorizations, authentications, programming languages, and furthermore features to be discussed in later studies. In summary, this thesis report has several contributions in resonating a valuable insight to the researchers in the supply chain management, automation, and blockchain technology and its use cases in SCM.

1.5 Report Organization

The preceding chapters entitles further details of the research on various aspects of the automation and blockchain technologies, methodology, and preliminary work. Chapter 2 provides an in-depth examination of smart contracts and blockchain platforms. It begins with an overview of key blockchain technologies, including Ethereum, Hyperledger Fabric, Quorum, and Corda, each with its own unique features and applications. The chapter then summarizes the critical aspects of these technologies, highlighting their relevance and differences. It further explores the application of blockchain in supply chain management, emphasizing its potential to enhance transparency and efficiency. Finally, the chapter reviews several companies—IBM Food Trust, Provenance, Walmart, and TradeWaltz—that have successfully integrated

CHAPTER 1

blockchain technology into their supply chain operations, showcasing real-world implementations and benefits.

Chapter 3 details the system methodology and approach used in the project. It begins with an outline of the system requirements, specifying both hardware and software/tool needs. Chapter 4 covers the system design, including an overview of Hyperledger Fabric, its transaction workflow (Execute, Order, Validate), and the supply chain architecture. It also introduces the Hyperledger Firefly development stack. While Chapter 5 details the implementation process, including software setup, deployment on the Kaleido platform, and implementation using an Ubuntu Linux Virtual Machine. It also covers generating networks, invoking smart contracts, and initializing the Hyperledger Firefly CLI.

Chapter 6 evaluates the system through testing and performance metrics, detailing the setup and results. It discusses the challenges encountered during the project and evaluates whether the objectives were met. The chapter concludes with a summary of findings, and the last chapter provides a final conclusion based on the overall project findings and offers recommendations for future work or improvements. This chapter summarizes the key findings and insights gained from the literature review, methodology, and preliminary work conducted throughout the project. It will highlight the significance of the research in advancing understanding and application of automation and blockchain technology in the SCM.

Chapter 2

Literature Review

2.1 Background to Smart Contracts, RPA and Blockchain Technologies

Smart contract is a type of automation that could be applied within the system architecture to boost overall performance, transparency, scalability, and security. Smart contracts are digital contracts that are made up of sets of bytecode instructions run in order [23]. Meanwhile, it can be utilized in storing and managing metadata and relevant to supplier onboarding and relationship management, including supply chain specifics (product data and related actions), quality certifications, and endorsements from food stakeholders [46].

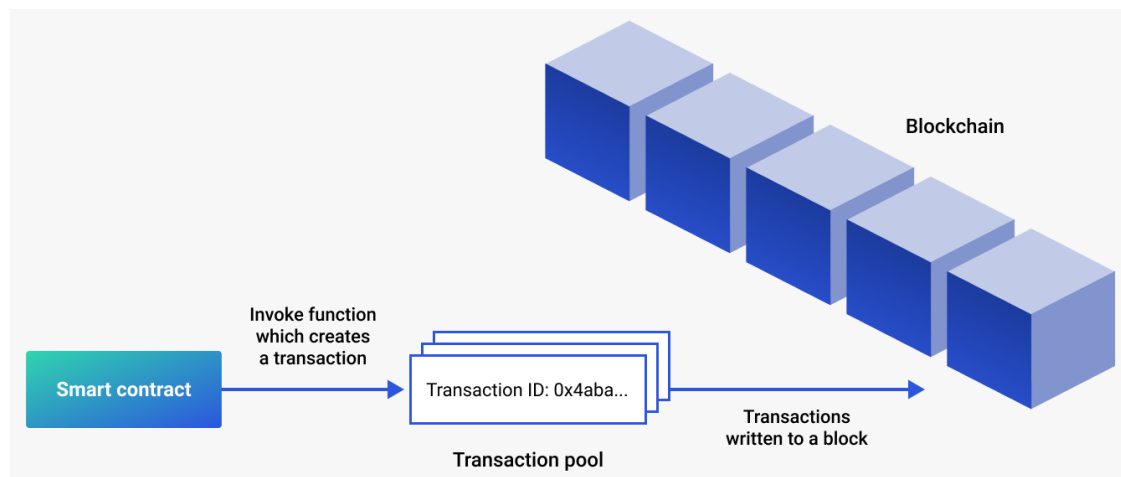


Figure 2.1.1 Structure of Smart Contract

Figure 2.1.2 shows when predetermined terms and conditions are met, the transaction is submitted onto the chain to allow the peers within the network to validate and execute it automatically [30]. When the transaction is completed, it will be updated to the chain and the action is immutable and irreversible [31]. No one is not allowed to change the state once it is written to the chain, and hence, provides transparency and traceability.

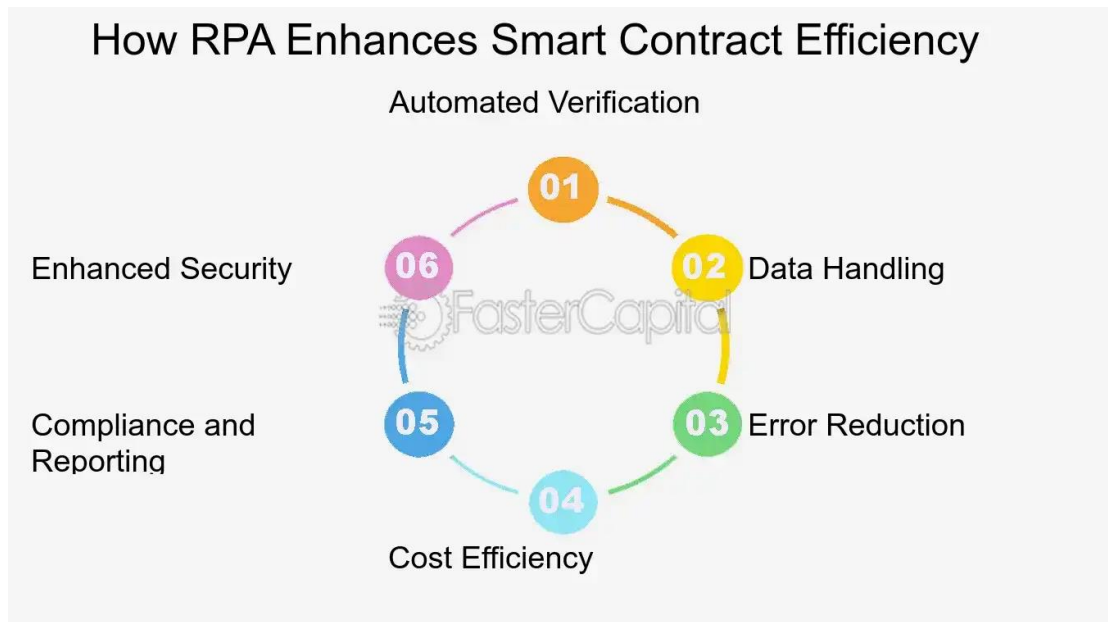


Figure 2.1.2 How RPA Enhances Smart Contract Efficiency

Smart contracts and RPA are able to work together to proceed with the automation within a digital environment. Smart contracts and RPA complement each other in automating complex workflows, where smart contracts ensure trust and enforceability while RPA handles the routine tasks associated with those agreements. By combining RPA's speed in handling tasks and smart contracts' automated enforcement, businesses can achieve end-to-end automation with faster and more reliable outcomes. Apart from that, both RPA and smart contracts further enhances the accuracy of data and ensure immutability by the automating of data entry and retrieval, and securely enforcing agreements. When integrated, they can automate and scale complex, multi-step workflows that involve both contractual obligations and business processes.

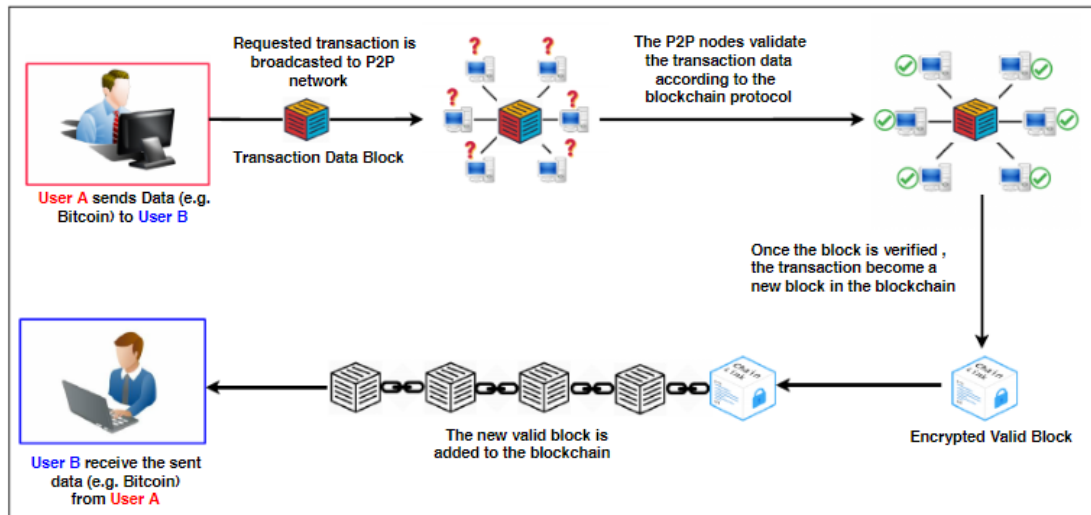


Figure 2.1.3 Smart Contract Execution

Ethereum, Hyperledger Fabric, Corda are among the most popular and widely used blockchain platforms and technologies by industries in today's era. Ethereum is a permissionless public blockchain-based platform that allows users to build decentralized and create smart contracts. On the other hand, Hyperledger Fabric is a permissioned, shared ledger. It allows a transaction of one ledger to discover and utilize the transaction and smart contracts on another ledger [14]. Quorum is a decentralized platform based on Ethereum that allows for private transactions and enables the giving of rights to manage network access. Corda allows users to record and manage agreements internally by providing them security, consistency, reliability, privacy, and authority [14].

2.1.1 Ethereum



Figure 2.1.1.1 Ethereum

Ethereum uses a high-level language, Solidity, a well-known and easy to understand built-in programming language in executing smart contracts on the platform [14]. The network is composed of nodes that maintain the integrity of network by validating transactions and ensuring consensus [13]. The Proof of Work (PoW) consensus algorithm, similar to Bitcoin's 'hash function', is initially employed by Ethereum. To ensure transaction validation and block addition to the blockchain, miners are engaged in a competitive endeavor to resolve complex mathematical riddles [13]. Nonetheless, Ethereum is now upgrading to Ethereum 2.0 network and transitioning to a PoS consensus mechanism.

Node authentication	N\A (permissionless, authenticated communication uses ECC)
Node roles granting	N\A (no roles are defined)
Transaction sender authentication	key-based
Transaction sender authorization	account owner does not require specific authorization
Transaction receiver authentication	N\A (all transactions are public)
Transaction receiver authorization	N\A (all transactions are public)
General remote user authentication	non-existing
Account owner remote authentication	passphrase-based
Remote user authorizations	non-existing, available methods depend on enabled modules

Table 2.1.1.1 Summary of Ethereum Authentication and Authorization Schemes

According to [17], in terms of Transaction Sender Authentication (TSA), the account owner does not need specific authorization to invoke any transaction if the private key has been proven (authentication). To confirm the sender of the transaction's identity, it employs recoverable ECDSA signatures in conjunction with key-based authentication [17]. Additionally, the passphrase that is received by unlocking the account via the *unlock_account* method serves as the basis for authentication for actions linked to the wallet. Its parameters include an address, a passphrase, and an optional time for signing transactions and sending money without reentering the passphrase.

CHAPTER 2

Smart contracts in Ethereum operate in the Ethereum Virtual Machine (EVM), which allows the contract code to trigger data reads and writes, not to mention execute computations such as cryptographic primitives and send messages to other contracts [14]. The bytecode also allows jumps, which makes the behavior Turing complete. When a Solidity contract is compiled, it is turned into a list of opcodes, which are operation codes and are recognized by shorthand [23]. The key component of the smart contract deployment is to generate the account address.

In this case, Ethereum uses the *Keccak-256 hashing algorithm*, a variant of the SHA-3 cryptographic hash function. The Ethereum address is formed by taking the rightmost 160 bits (20 bytes) of the hash value and generating it from the resulting public key. On the Ethereum blockchain, this address serves as the smart contract's unique identifier [17]. The elliptic curve *secp256k1* is used to generate the pair of keys, and each identity in the system is given a distinct account address. Upon initiation of a transaction, the sender's identity is verified by the transaction signature to ensure that the derived account address corresponds exactly to the one recorded in the global network state [17].

The equation for generating the Ethereum address from the Keccak-256 hash value is as follows:

$$\text{Address} = \text{rightmost_160_bits}(\text{Keccak} - 256(\text{Bytecode}))$$

Where:

- *Keccak – 256(Bytecode)* represents the output of the Keccak-256 hashing algorithm applied to the bytecode of the smart contract.
- *rightmost_160_bits* function extracts the rightmost 160 bits (20 bytes) from the hash value to obtain the Ethereum address.

This algorithm ensures that each deployed smart contract on the Ethereum blockchain has a unique address derived from its bytecode, providing a secure and deterministic method for addressing and interacting with contracts on the network.

2.1.2 Hyperledger Fabric



Figure 2.1.2.1 Hyperledger Fabric

Hyperledger Fabric, also known as Fabric, is one of the projects under the Linux Foundation in the December 2016 and the first distributed OS for permissioned blockchains [25]. For example, Go, Java and Node.js. Hyperledger Fabric is an open-source platform and technology used in industrial enterprise solutions. It leverages the Practical Byzantine Fault Tolerance (PBFT) approach to attain agreement, as well as a combination of Node.js and JavaScript [24]. PBFT makes use of cryptographic techniques like hashing, signature verification, and signatures to make sure that nothing can be changed and that the evidence is unquestionable. As for the secure hashing algorithm, Fabric uses SHA-256 for hashing. A cryptographic hash function called SHA-256 (Secure Hash Algorithm 256-bit) generates a 256-bit (32-byte) hash value [27]. Since the hash of one block is determined using the hash of the preceding block and transaction, this enables the block to be linked to the chain structure.

Message sender authentication	Certificate-based
Node roles granting	ABAC
Transaction authorization	ACL
Transaction receiver authentication	N\A (all transactions are accessible by channel members)
Transaction receiver authorization	N\A (all transactions are accessible by channel members)
General remote user authentication	Certificate-based
Remote user authorizations	ABAC

Table 2.1.2.1 Summary of Hyperledger Fabric Authentication and Authorization Schemes

Chaincode, in the context of blockchain technology, is a term used in Hyperledger Fabric, and are often referred to as smart contracts [26]. It is a program code that implements application logic and executes transactions on the blockchain network to move digital assets in line with established rules [27]. Thus, the business logic layer of Fabric is made up of chaincode, which establishes the guidelines and practices for handling and validating transactions. Once a chaincode has been deployed and the contract's requirements have been met, the code executes and automatically completes the transaction [27]. It is critical in ensuring that blockchain networks operate efficiently, safely, and in accordance with the unique requirements of an application.

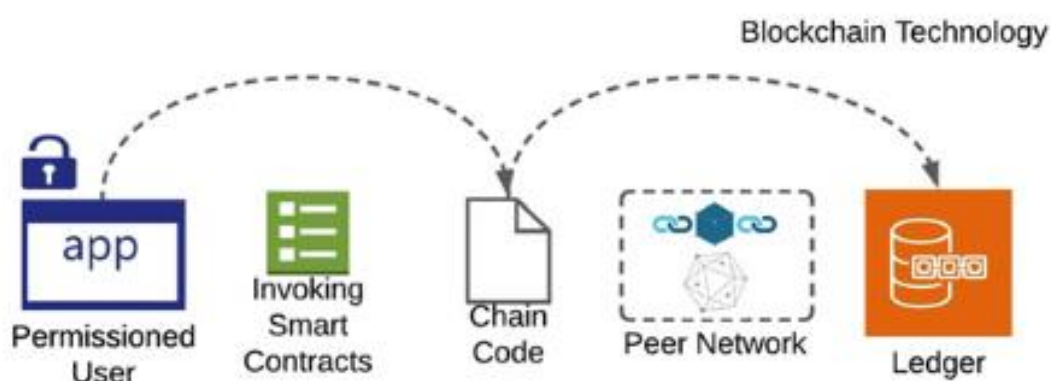


Figure 2.1.2.2 Blockchain Technology for Smart Contracts

Fabric, in terms of transactions, comprises of two types, namely Deploy and Invoke transactions. Deploy transactions is the creation of new smart contracts and deploy the transaction for execution. Meanwhile, invoke transactions will run the operation in regards of previous deployed chaincode. The deployment of transaction does not require specific authorization scheme since everyone has the access to transactions within the same channel [14].

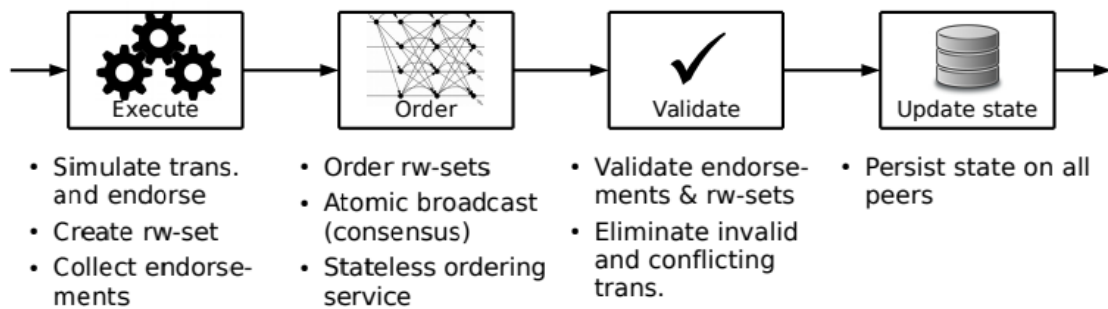


Figure 2.1.2.3 Execute-order-validate architecture of Fabric

Fabric uses architecture which is *execute-order-validate* paradigm by dividing the transaction flow into three stages. The first stage is the execution stage where the transaction is executed by peers and checked for the correctness [22]. As for the ordering stage, it ensures that the consensus protocol is sent and on the order of transactions in the block. Lastly, the validation stage, where the transaction is validated by each and every peer within the network aligning with specific trust assumptions. The block will be added on the blockchain if the transactions are legitimate [28]. When the state of any of the Fabric network nodes changes, they are all refreshed and monitored simultaneously [24]. With the Hyperledger Fabric network, transactions may be completed almost instantly and are globally accessible.

2.1.3 Quorum



Figure 2.1.3.1 Quorum

As Quorum is derived from Ethereum, they are likewise having the same structure. The fundamental purpose of Quorum is to employ cryptography in order to restrict access to sensitive data exclusively to the individuals involved in the transaction. The method involves using a cohesive blockchain and incorporating smart contract software architecture, as well as making changes to the Ethereum platform. The smart contract design allows for the separation of sensitive data into discrete sections, which facilitates improved privacy and security measures. This is accomplished by modifying the go-ethereum codebase, which includes changes to both the block proposal and validation procedures, protecting the integrity and security of sensitive data. [29]

Message sender authentication	key-based, optionally certificate-based if TLS CA enabled
Node roles granting	N\A (no node roles defined)
Transaction sender authentication	key-based
Transaction sender authorization	account owner does not require specific authorization
Transaction receiver authorization	ACL
General remote user authentication	non-existing
Account owner remote authentication	passphrase-based
Remote user authorizations	non-existing, available methods depend on enabled modules

Table 2.1.3.1 Summary of Quorum Authentication and Authorization Schemes

On top of that, Quorum employs the Ethereum Geth node to handle personal transactions, and similar to Ethereum, uses key-based authentication with recoverable ECDSA signatures. The objective is to authenticate that the sender is the rightful owner of the account from which the transaction originated [17]. In order to grant authorization to a new node, the designated public key must be included in the list of keys for each node. In contrast, revoking a node necessitates the removal of each node from the list. Quorum has restricted access to private transactions to a set number of nodes determined by the sender [17]. Despite the similarity between Quorum and

CHAPTER 2

Ethereum, there are some differences. The aforementioned states that Quorum is permissioned, and not open to all [15]. Only those who have granted access and validated can be involved in the network.

During the modified block validation process of Quorum, all nodes meticulously validate both public and private transactions by executing the contract code linked to those transactions. However, in the case of other "private transactions" where a node is not directly involved, the node will bypass the contract code execution process. This approach guarantees that nodes only execute contract code for transactions that are pertinent to them, maintaining efficiency and privacy within the Quorum network. Quorum efficiently manages private data and ensures transaction integrity and network efficiency by selectively executing contract code based on transaction involvement.

On top of that, Quorum network operates on a voting-based consensus algorithm [29] known as QuorumChain, where the voting rights are assigned to each and every member within the network via smart contracts, and the status is then tracked using the tally votes. The block that receives the highest number of votes will emerge as the winner and is accepted as the authoritative head of the chain. The election results are then verified and audited to achieve the prevention of fraud and safeguard the accuracy of the results [16].

2.1.4 Corda



Figure 2.1.4.1 Corda

R3 Corda enables customers to build their smart contract-driven applications using the Java and Kotlin programming languages [32]. It provides users two options to adapt to; use the open-source code to deploy personalized network or pay to use the available Corda network.

Message sender authentication	Certificate-based
Node roles granting	ABAC
Transaction sender authentication	Relies on certificate-based node authentication
Transaction authorization	Provided by notaries node
Transaction receiver authorization	N/A (the transactions are only sent to authorized nodes and not broadcast)
General remote user authentication	Password-based
Remote user authorizations	Capability list

Table 2.1.4.1 Summary of Corda Authentication and Authorization Schemes

In Corda, a compatibility zone is designed for the permissioned network. [14] states that the network comes in four types of certificate authorities (CA) including the root network CA, the doorman CA, the node CAs, and the legal identity CAs. The doorman CA plays the role of intermediary, while each node in the node CAs serve its own CA. Last, the legal identity CAs are capable of signing transactions and issue certificates for confidential legal identities. There are two types of notaries used in transaction validation. The initial notary is assigned to the 'validating' and 'non-validating' categories, where users are granted access to all transactions and verify their consistency without having access to the transaction content. However, they are notified that the input states have not yet been consumed.

Corda has implemented the certificate-based authentication using digital signatures in authentication of the message sender. Moreover, Due to the fact that each transaction is associated with a node, Corda treats them as standard messages; therefore, transaction sender authentication is handled in the same manner as message sender authentication

CHAPTER 2

[14]. All transactions are confidential and only shared among stakeholders. Notaries offer authorization to trigger a transaction. In addition, password authentication, controlled at the node level, facilitates remote user authentication. To connect remotely to a node, users must give proper credentials, such as a username and password.

Corda adopts a distinct methodology when it comes to smart contracts. The new state can be generated by any arbitrary code and there is no requirement to distribute it. Therefore, the code responsible for generating transactions might be owned exclusively by a certain entity. The purpose of the contract code in Corda is to verify the transaction proposal. In other words, a Corda contract is a validation function that enforces limitations on how a transaction can alter states. The validation function is disseminated to all nodes on a need-to-know basis and is executed by all relevant parties to authenticate a transaction.

Due to the requirement for the validation function to consistently generate the same output for a given input, there are constraints on what may be done with the contract code. Corda utilizes a simplified version of the Java Virtual Machine (JVM) to ensure consistent and predictable execution of contracts. For instance, the utilization of stochastic number generators in the contract code would yield disparate outcomes for identical inputs and is thus prohibited. The same principle applies to citing external data that may undergo changes in the future. [33]

CHAPTER 2

2.2 Summary of Reviewed Blockchain Technologies

<i>Features</i>	<i>Ethereum</i>	<i>Hyperledger Fabric</i>	<i>Quorum</i>	<i>Corda</i>
<i>Node Authentication</i>	N/A (permissionless, authenticated communication uses ECC)	Message sender authentication (Certificate-based)	N/A (no node roles defined)	Message sender authentication (Certificate-based)
<i>Node Roles Granting</i>	N/A (no roles are defined)	ABAC	N/A (no node roles defined)	ABAC
<i>Transaction Sender Authentication</i>	Key-based	Key-based	Key-based	Relies on certificate-based node authentication
<i>Transaction Sender Authorisation</i>	Account owner does not require specific authorisation	Account owner does not require specific authorisation	Account owner does not require specific authorisation	Provided by notaries node
<i>Transaction Receiver Authentication</i>	N/A (all transactions are public)	N/A (all transactions are accessible by channel members)	N/A (all transactions are accessible by channel members)	N/A (transactions only sent to authorised nodes and not broadcast)
<i>Transaction Receiver Authorisation</i>	N/A (all transactions are public)	ACL	N/A (all transactions are accessible by channel members)	N/A (transactions only sent to authorised nodes and not broadcast)
<i>General Remote User Authentication</i>	Non-existing	Non-existing	Non-existing	Password-based

CHAPTER 2

<i>Account Owner Remote Authentication</i>	Passphrase-based	Non-existing	Passphrase-based	Non-existing
<i>Remote User Authorisations</i>	Non-existing, available methods depend on enabled modules	Non-existing	Non-existing	Capability list
<i>Smart Contract Execution</i>	Ethereum Virtual Machine (EVM)	Docker containers	Ethereum Virtual Machine (EVM)	JVM-based
<i>Smart Contract Algorithm</i>	SHA-3-256 hashing algorithm (Keccak-256)	PBFT	Modified block validation procedures	Deterministic validation function
<i>Programming Language</i>	Solidity	Go, Java, JavaScript, TypeScript	Solidity	Java, Kotlin
<i>Transaction Finality</i>	Eventual	Final	Eventual	Final
<i>Privacy</i>	Public	Permissioned	Private, Permissioned	Confidential Transactions
<i>Key Features</i>	Decentralisation, Smart Contracts	Scalability, Permissioned, Modularity	Privacy, Permissioned, Performance	Privacy, Compatibility, Security
<i>Consensus Mechanism</i>	PoW transitioning to PoS	PBFT, Raft	Voting-based consensus mechanism (QuorumChain)	Raft, Notary

2.3 Blockchain in Supply Chain Management

Supply chain management (SCM) is crucial in today's global economy for it exerts a substantial impact on the worldwide economy [42]. In today's supply chain, there is no secure method in place within the present supply chain to store confidential data of consumers [41]. Blockchains play a crucial role in facilitating agreements, payments, and delivery in contemporary supply-chain management systems. Within the supply chain context, a customer and a supplier establish a contractual agreement whereby the client agrees to swiftly transfer funds to the supplier's account upon receiving a product that meets established specifications and quality standards. This arrangement can be executed as a smart contract, wherein cash will be automatically transferred from the consumer to the supplier's blockchain wallet, akin to a bank account, as soon as the product is received in flawless condition, meeting all set requirements. The blockchain is a cutting-edge technology that has the capacity to greatly improve supply chain management due to its inherent security, heightened transparency, traceability, and efficiency [40].

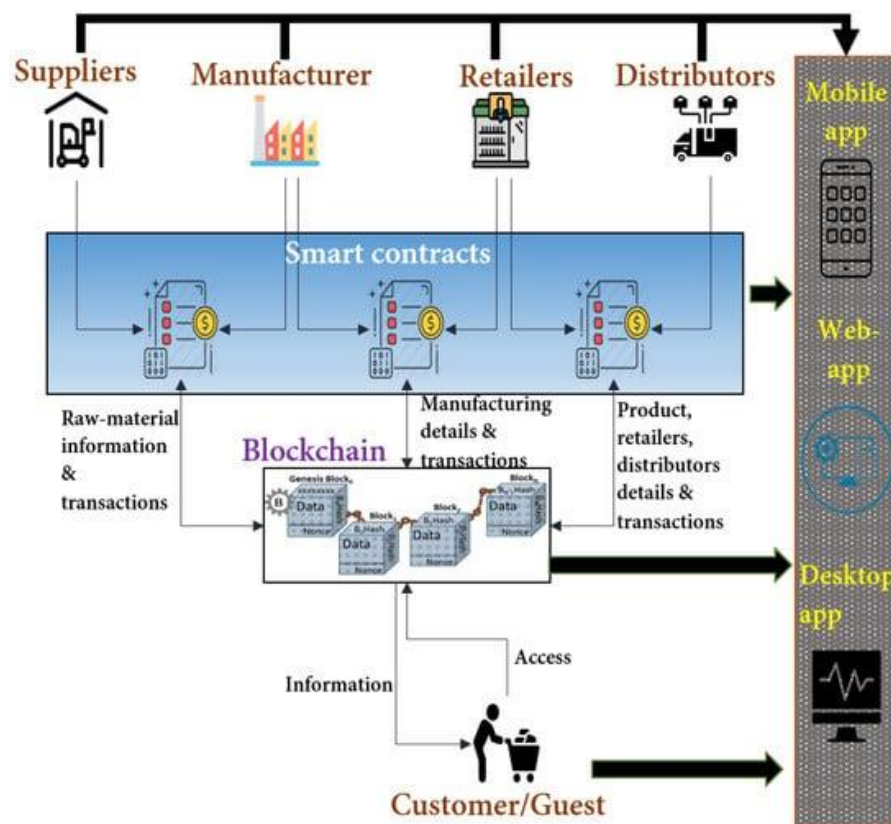


Figure 2.3.1 Blockchain-based Supply Chain Management

The majority of current tracking and traceability systems, commonly employed by supply chain networks, encounter issues related to centralized management and data protection. Blockchain (BC), a core technology, has garnered significant attention from both academia and businesses in recent years because to its inherent characteristics of immutability and decentralization. Implementing BC technology in the supply chain will revolutionize current supply chain systems and eliminate reliance on third-party systems. It has the ability to overcome a wide range of challenges in the supply chain. The promise of this technology to transparently and securely trace all types of transactions motivates us to explore the opportunities that blockchain offers throughout the supply chain. [40]

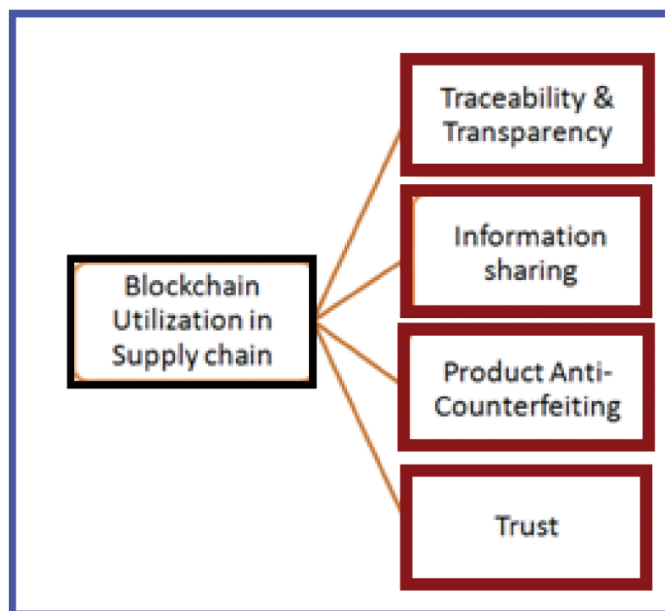


Figure 2.3.2 Blockchain Utilization in Supply Chain

The primary objective and difficulty of efficient SCM is to enhance the performance of the supply chain by considering exterior factors like customer satisfaction and internal factors like operational efficiency. This involves analyzing and eliminating any wastage, issues, or complications inside the internal supply chain. The integration of SC modules requires a strong commitment to Blockchain technology, a high level of confidence, collaborative decision-making, and the exchange of accurate information. There are lots of notable applications of blockchain technology being applied in the SCM. To name a few, Food SC, Pharmaceutical SC, Automotive SC, Electronics SC, Finance SC, and Logistics SC [43].

CHAPTER 2

It is found that automatic data validation provided by BC reduces the necessity for double-checking in the SC, as well as offering tracking and tracing capabilities. BC enables transaction monitoring by offering verifiable evidence of transaction origin. Blockchain technology facilitates increased scalability and enhanced data precision, resulting in accelerated end-to-end supply chain processes. BC-enabled applications facilitate the rapid distribution of data over the whole network within seconds. Consensus approaches offer a reliable and logical basis for smart contracts, allowing for supply chain automation and operational advantages. In the context of SCM, blockchain technology has the potential to enhance transparency, traceability, efficiency, and data security. [41]

2.4 Use case of Blockchain Based SCM

2.4.1 IBM Food Trust

The global food supply chain has grown in complexity as a result of the widespread availability of food products from around the world, regardless of season or location. This complexity has increased the demand for trust and openness throughout the supply chain. IBM Food Trust addresses this difficulty by ensuring transparency across the whole food ecosystem, from farm to consumer. IBM Food Trust is a blockchain-driven network that connects key players in the food supply chain, including farmers, processors, distributors, and retailers [49]. Utilizing Hyperledger Fabric enhances the traceability, transparency, and overall efficiency of supply chain operations. This solution allows involved parties to share and access their food supply chain data through a shared, secure, immutable record of food provenance and transactions [48].

By enabling end-to-end product tracing, IBM Food Trust helps businesses share documentation securely and build consumer trust. Besides, these platforms minimize human error and create a more reliable and transparent supply chain through automating manual processes. The manipulation of blockchain technology in the food supply chain eventually allows authorized users to quickly access actionable data across the entire SC, initially from farm to retail store and ultimately to the consumer. It enables users to retrieve comprehensive information about any food item, including its complete history, current location, and related data such as test results, certifications, and temperature records, within seconds. [48]

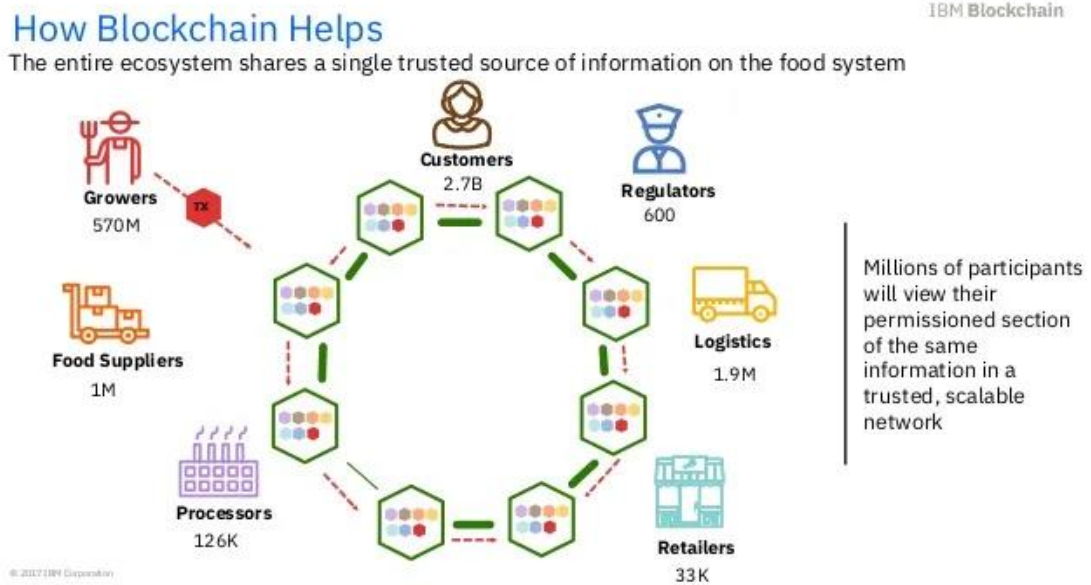


Figure 2.4.1.1 Blockchain Role in Supply Chain

IBM Food Trust merges supply chain modules with core blockchain features, integrating technology, standards, interoperability, and governance to generate economic value within the food ecosystem. Food Trust leverages blockchain's decentralization to enable multiple participants to share food-related data on a permissioned blockchain model. Each blockchain node is managed by a different entity, ensuring that all data is encrypted, inerasable, and trustworthy. They can only have access to data they are authorized to view.

Hyperledger Fabric, the blockchain framework underpinning Food Trust, is distinct from open-source platforms like Ethereum. Unlike Ethereum's permission-less network, where data is accessible to all participants, Hyperledger Fabric's permissioned structure offers greater data privacy and flexibility, allowing companies to maintain control over data access on the blockchain network. [47]

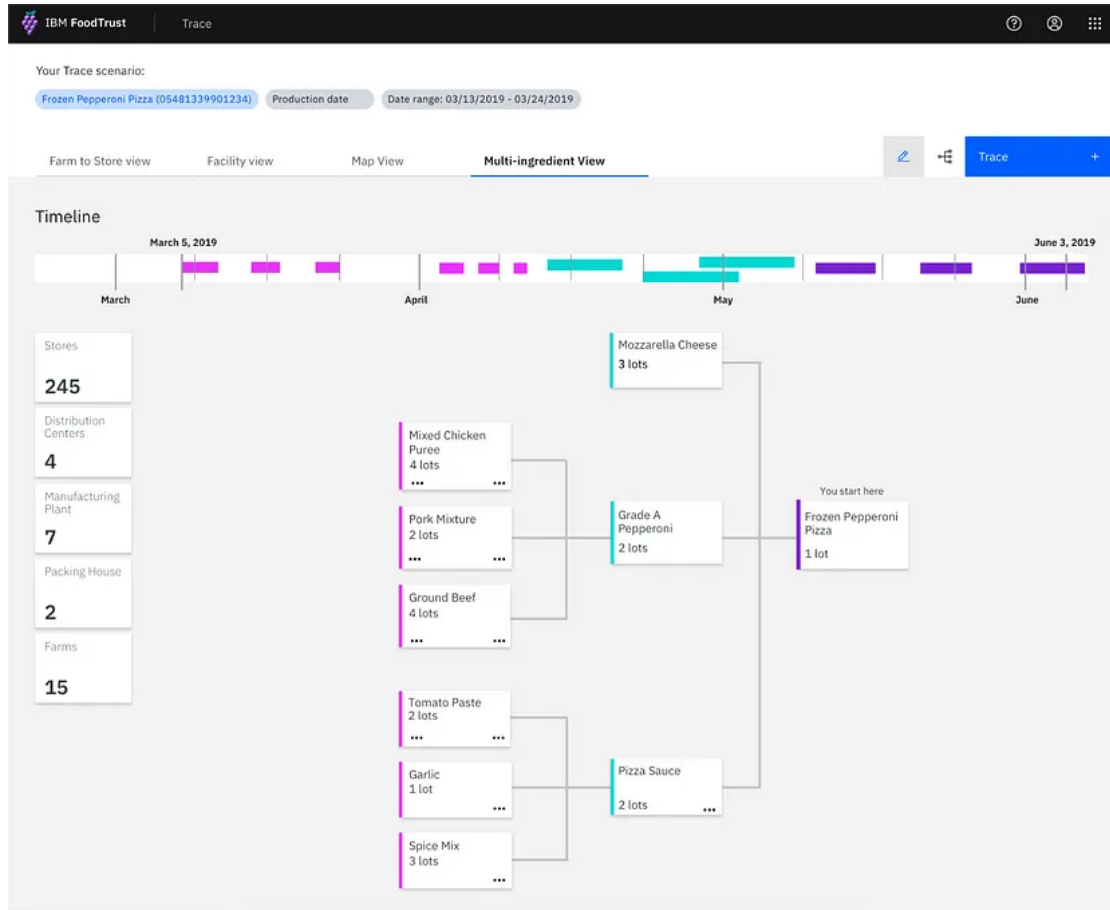


Figure 2.4.1.4 IBM Food Trust Trace Module (b)

- b. **Fresh Insights Module:** This module connects various product data points to provide visibility into inventory across the supply chain. It tracks the time since production and until expiration, helping to identify at-risk inventory. Suppliers can use this information to improve product freshness, reduce losses, and identify inefficiencies.

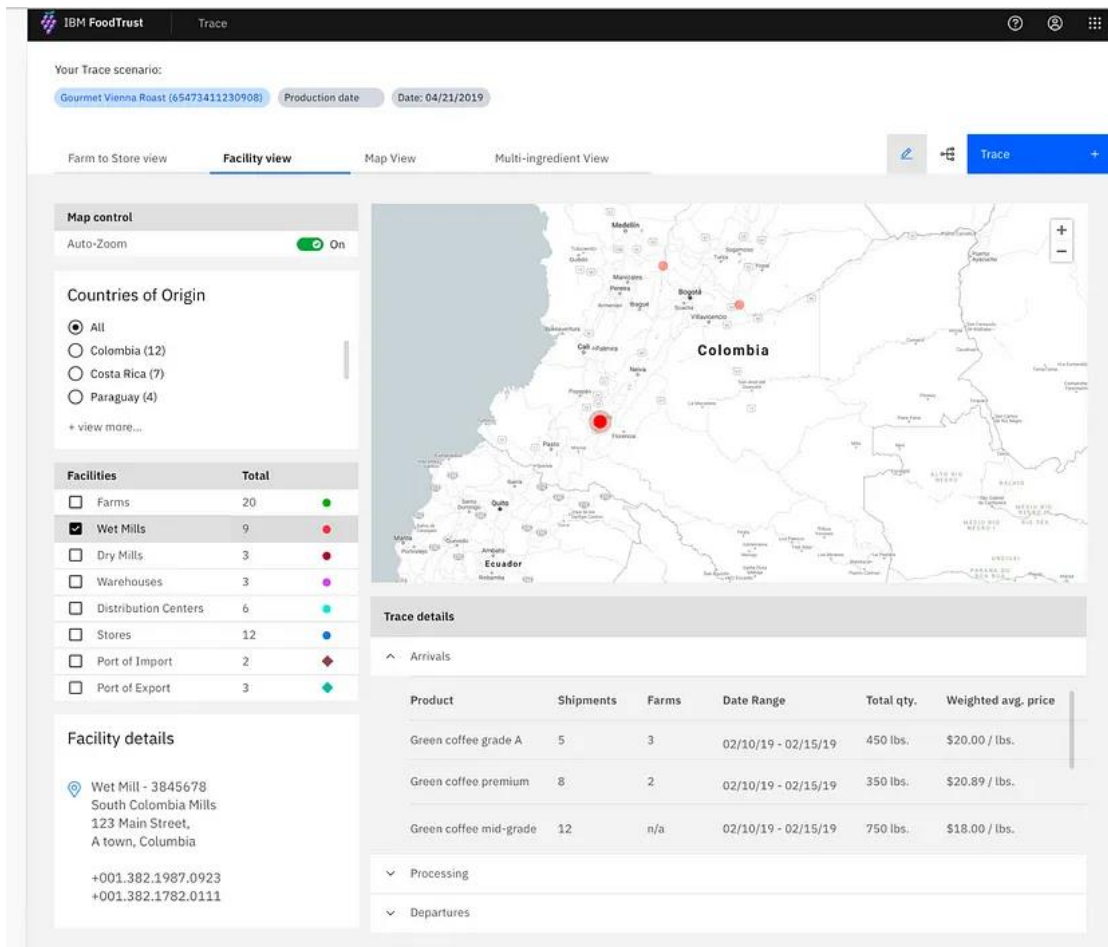


Figure 2.4.1.5 IBM Food Trust Fresh Insights Module

- c. **Certifications Module:** This module digitizes crucial certificates and inspection documents, optimizing the efficiency of information management. It helps certify the provenance of products and ensures their authenticity.

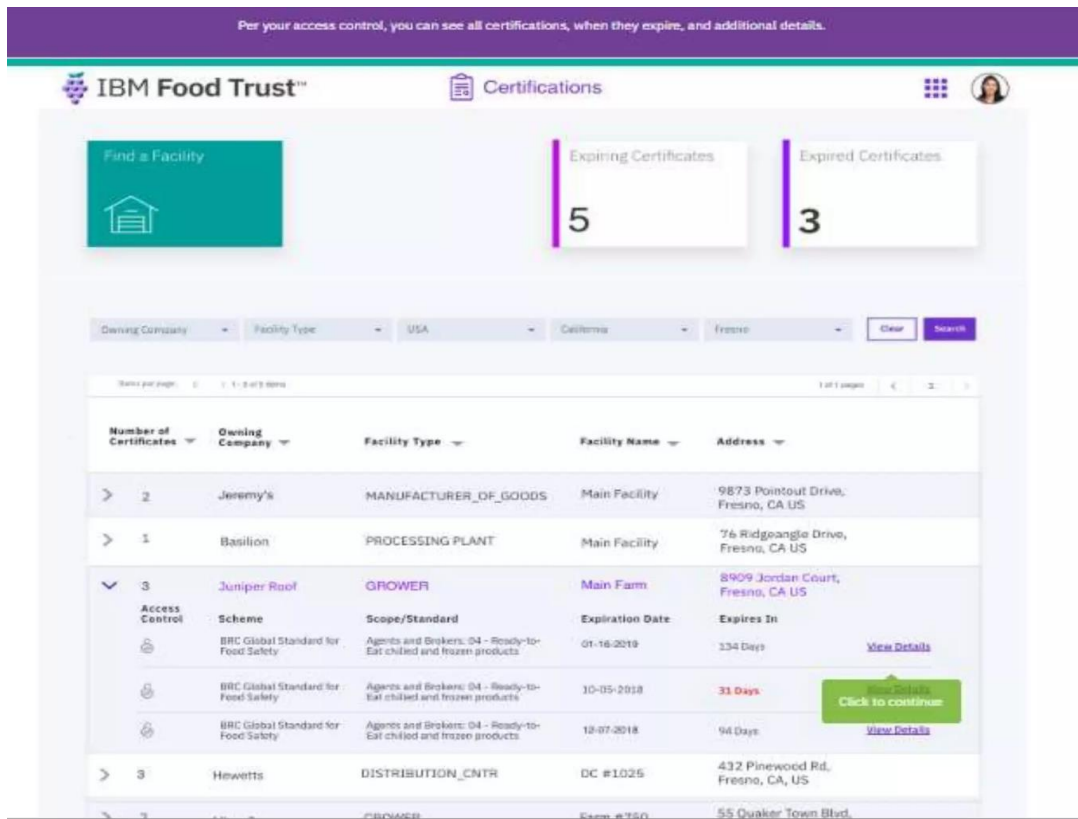


Figure 2.4.1.6 IBM Food Trust Certification Module (a)

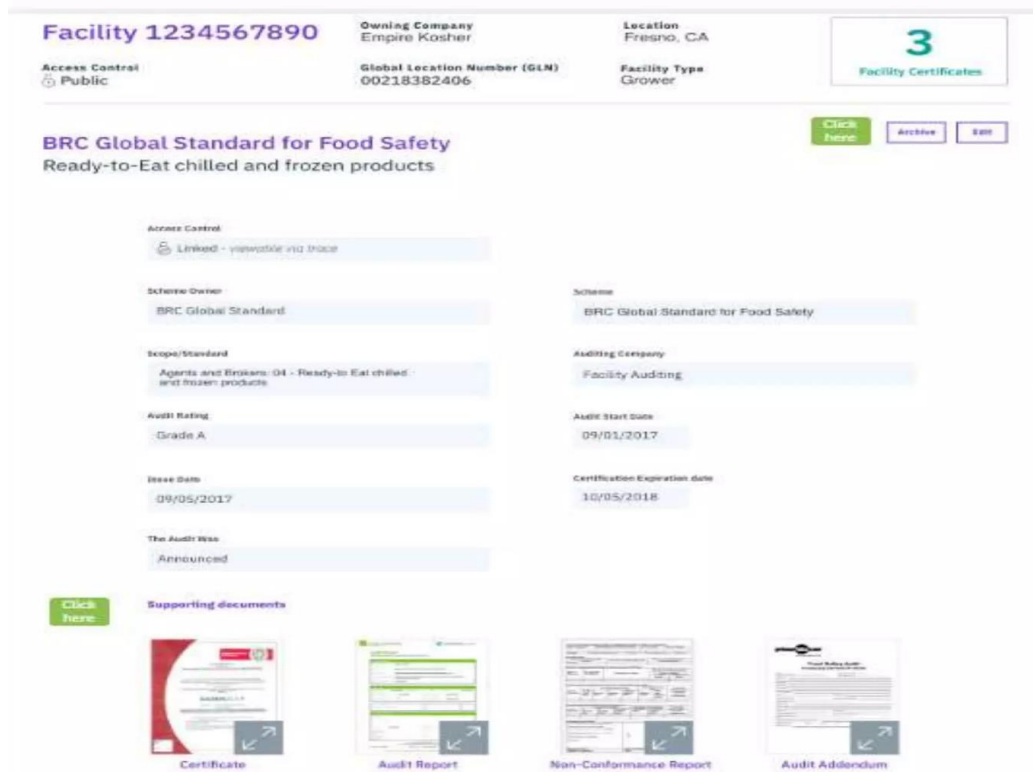


Figure 2.4.1.7 IBM Food Trust Certification Module (b)

- d. Consumer App:** IBM Food Trust aimed to offer a customizable consumer app that would allow in-store customers to scan product QR codes and barcodes. This program would allow users to explore and verify the origins and journeys of any food item, including its contents. The purpose was to provide consumers with transparency regarding the food goods they buy, so encouraging trust and informed decision-making.

2.4.2 Provenance

Provenance is a data-driven platform that verifies and highlights sustainability credentials for consumer-packaged goods. The platform allows businesses to communicate their sustainability efforts and impact transparently, providing consumers with valuable information from the point of discovery through to the checkout process [53]. Provenance's system is designed to improve transparency within the supply chain by ensuring that all participants meet certification and standards requirements.

The framework comprises six measured components: registering, standards, generation, manufacturing, tagging, and user-facing modules. Each of these modules works freely but is coordinated into a bound-together blockchain, making a cohesive environment inside the framework. The blockchain records exchanges by putting away information in a shared open record, which permits for auditability. Moreover, smart contracts are utilized to encourage operations inside the chain, counting money-related and data trades. [48]

A case study about tuna tracking and certification was conducted over half a year in Indonesia, collaborating with the NGO Humanity United and the International Pole and Line Foundation (IPNLF) to track and trace yellowtail tuna [48]. In this case, Provenance implemented a system consisting of data interoperability to ensure secure, continuous and accessible tracking and tracing of items and certifications across the SC. This system introduced a point of sale (POS) for tracking smart tags. Fishermen, for instance, could enroll their catch by sending an SMS, which would make a new block within the blockchain with a lasting and special ID. This interesting ID was then physically joined to the fish utilizing technologies like QR codes or RFID labels. As the fish moved to the supplier, the digital asset followed, with each transaction recorded on the blockchain. The fishermen's identities were also stored, allowing them to track and trace the fish using public software libraries that connect with the blockchain. The picture below depicts Tuna's provenance, which is supported by blockchain technology and available to participants throughout the supply chain.

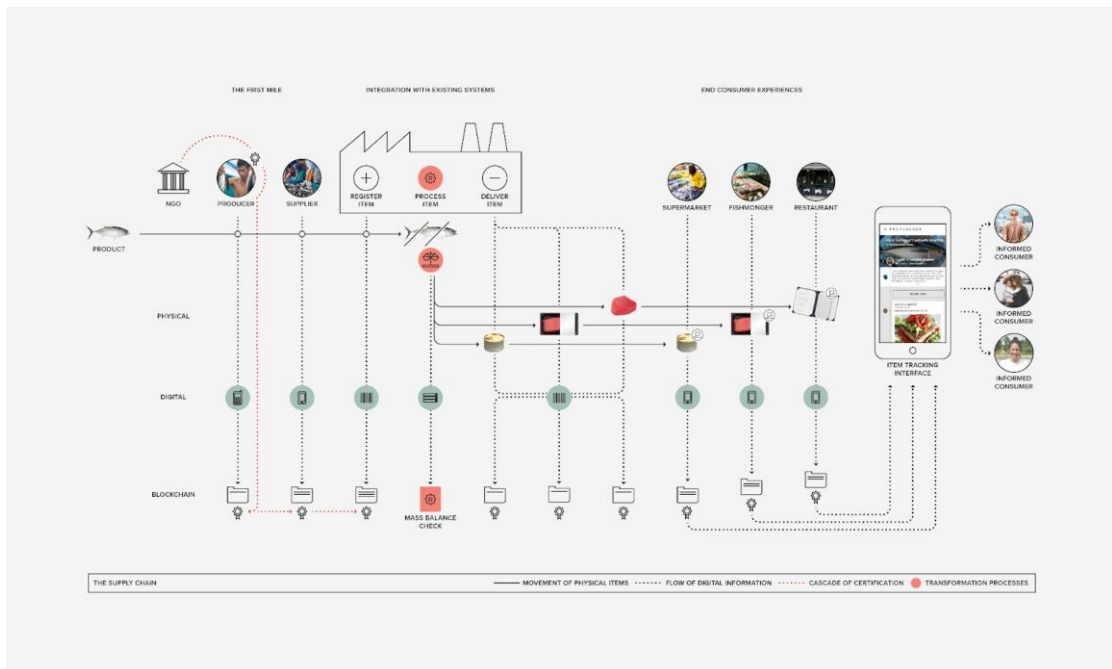


Figure 2.4.2.1 Provenance of Tuna

2.4.3 Walmart

Walmart is one of the world's largest retailers, with an extensive and complicated supply chain that has expanded to include 210 distribution channels. Its success is due to its commitment to driving innovation and outstanding operations in the supply chain, being well-known for its efficiency, scale, and ability to serve millions of customers daily [58]. In this scenario, Walmart uses IBM's blockchain-based Food Trust to track food products from farm to store shelves. This helps ensure the freshness and safety of perishable items. By scanning product information onto the blockchain, Walmart can trace the origins and journey of food items, improving recall capabilities in case of contamination or safety concerns.

Walmart's use of blockchain technology has significantly improved its supply chain management, particularly in the areas of food safety, supplier management, and efficiency. One of the most impactful applications is the faster response to food safety issues. With blockchain, Walmart has drastically reduced the time needed to trace the origin of a food product from 7 days to just 2.2 seconds. This capability is essential during foodborne illness outbreaks or contamination incidents, allowing the company to swiftly isolate affected batches and remove them from shelves, protecting consumers. Apart from that, the adoption of blockchain technology increases transparency in terms of accessing detailed, immutable records about where products come from, who handled them, and how long they spent at each stage in the supply chain. This eventually increases food safety standards and builds trust with consumers. Suppliers can upload certificates of origin, safety audits, and other compliance documentation directly to the blockchain, simplifying the auditing process.

In terms of supplier management, Walmart requires certain suppliers, especially in fresh produce, to participate in its blockchain system. This ensures consistent data recording across the supply chain, improving communication between Walmart and its suppliers. As a result, supplier collaboration is enhanced, and supply chain optimization becomes more efficient. Moreover, blockchain technology aids Walmart in achieving cost reduction and waste minimization. By providing greater visibility into product expiration dates and quality, the company reduces food waste and optimizes inventory

CHAPTER 2

management, helping to avoid over-ordering or spoilage. Compliance and sustainability are also bolstered through Walmart's blockchain system. The technology offers verifiable, tamper-proof data that can be audited, ensuring that Walmart's supply chain adheres to industry regulations. Additionally, Walmart leverages blockchain to meet its sustainability goals by ensuring ethical sourcing practices. Furthermore, blockchain holds suppliers accountable for maintaining product quality and compliance standards. The transparent, shared data available to all participants in the supply chain allows Walmart to quickly identify inefficiencies or lapses, creating a more reliable system.

Walmart initially piloted blockchain technology with mangoes and pork and later expanded the effort to include leafy greens such as lettuce and spinach. As the company explores further scaling across various food categories and regions, blockchain technology continues to enhance its supply chain operations. Lastly, supplier integration into Walmart's decentralized blockchain network ensures that all stakeholders can securely share information without relying on a central authority, further solidifying Walmart's blockchain-powered supply chain. These blockchain initiatives help Walmart build a more resilient, transparent, and efficient supply chain, improving food safety, product quality, and sustainability while reducing costs and waste. [59]

2.4.4 TradeWaltz

TradeWaltz is a cross-industry trade information collaboration SaaS platform that provides security and integrated management of trade transaction data based on blockchain technology [55]. Launched in 2017, TradeWaltz aims to streamline communication and documentation processes across various industry sectors, including import/export, manufacturing, finance, insurance, logistics, and carriers. TradeWaltz aims to enhance international trade transactions' efficiency, transparency, and trustworthiness.

In this case, TradeWaltz enables the creation, management, and verification of digital trade documents on a blockchain ledger. TradeWaltz facilitates the generation and transmission of standard trade documents such as permits, certificates of origin, insurance policies, invoices, and shipping instructions. This digitalization reduces paperwork, speeds up processes, and minimizes the risk of fraud. It utilizes smart contracts to automate and enforce trade agreements. Smart contracts execute predefined conditions automatically, reducing the need for intermediaries and mitigating disputes. Meanwhile, with the use of blockchain technology, security is guaranteed, and users can track the status of shipments and transactions in real-time. This transparency helps stakeholders track progress and manage exceptions more effectively.

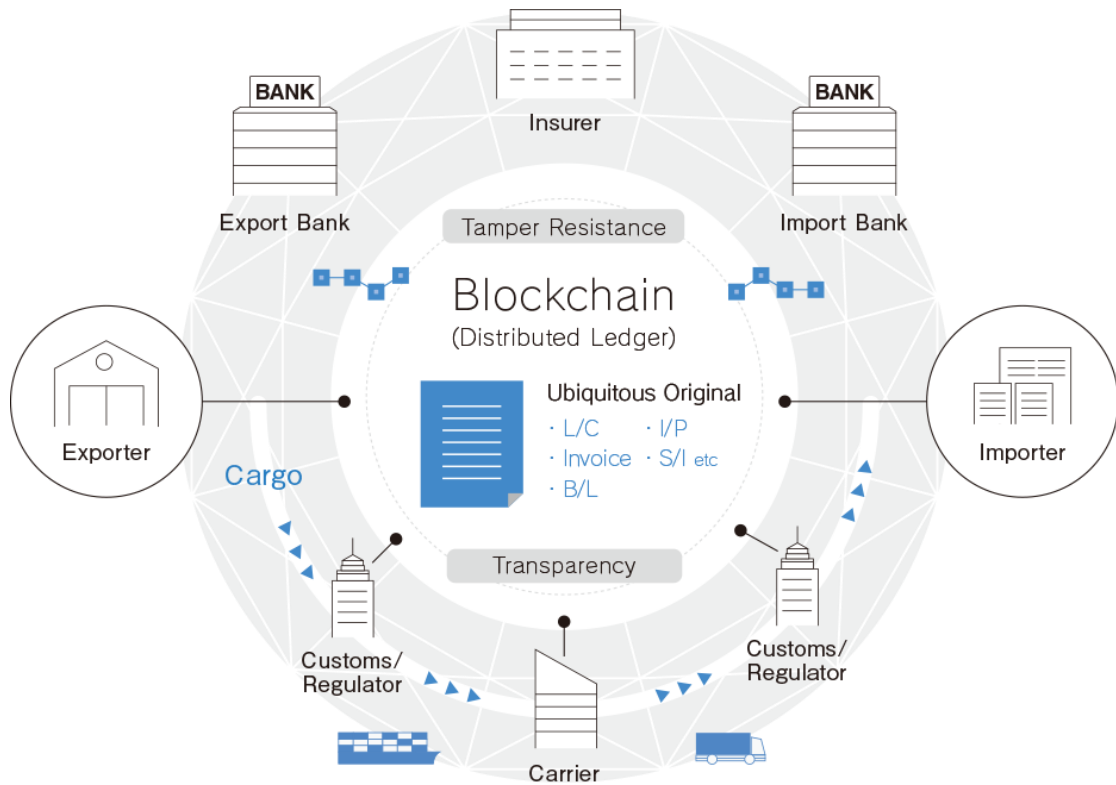


Figure 2.4.4.1 TradeWaltz Trade Ecosystem

The TradeWaltz platform architecture establishes a trade ecosystem that enables stakeholders in the sector to effortlessly communicate information using standard digital documents—without risking security breaches or exposure to other data. The platform covers a wide range of parties in the trade industry, allowing for seamless interactions between importers, exporters, manufacturers, financial institutions, insurance providers, and logistical businesses. They can generate documents and promptly send them to necessary parties, and document digitalization will result in minimal to no lag time in communication. [56] & [57]

Chapter 3

System Methodology/Approach

This chapter presents an outline of the suggested methodology and approach for the development of the smart contract framework project. The strategy and methodology are designed to utilize Hyperledger Fabric as the foundational blockchain infrastructure due to its robust features and compatibility with enterprise-grade blockchain applications. In this case, the Kaleido platform is used as the platform for the sandbox environment, and the project development is carried out using a laptop. Hyperledger Fabric stands out as a versatile and powerful platform, particularly suited for enterprise applications due to its permissioned nature, strong privacy features, and flexible architecture.

Secure Software Development Lifecycle (SSDLC) will be integrated into the project “RPA in Secure SCM”. SSDLC is an approach used in developing secure software and ensures all the processes are having security measures [44]. The objective of SSDLC is to ensure that security is an integral part of the software development process, in the end ensuring that final product is in a secure manner. It involves planning, analysis, design, development, testing, release, and maintenance processes [45].



Figure 3.0 SSDLC Process

Security requirements and objectives are to be defined in the planning phase. Meanwhile, in the analysis phase, security risk assessment needs to be conducted to understand the security context on which vulnerabilities relate to the selected resources. Moving on next to design which involves creating a secure software architecture for the project. During the development and testing phase, secure coding standards are required and perform various testing to identify the weaknesses in the designed code, and to solve vulnerabilities. In the release phase, secure deployments are of utmost importance to ensure the software is configured securely in the production environment. Lastly, ongoing maintenance and monitoring are needed to identify and address security issues.

3.1 System Requirement

3.1.1 Hardware

The hardware used in this development is laptop, which is the MSI GF63 Thin 9SCXR model, featuring an Intel Core (TM) i5-9300H processor. It runs Windows 11 and features an NVIDIA® GeForce® GTX 1650 graphics card with Max-Q Design. The system boasts 12GB of DDR4-2666 RAM for efficient multitasking capabilities and offers ample storage space with a 512GB SATA HDD. The platform used to test on the development is Kaleido, and it comprises of all the needed tools to develop the project.

Description	Specifications
Model	MSI GF63 Thin 9SCXR
Processor	Intel Core (TM) i5-9300H
Operating System	Windows 11
Graphic	NVIDIA® GeForce® GTX 1650 With Max-Q Design
Memory	12GB RAM DDR4-2666
Storage	512GB SATA HDD

Table 3.1 Specifications of Laptop

3.1.2 Software/Tools

Software/Tools	Description															
1) Kaleido	<p>Kaleido is a full stack SaaS platform [36] and offers Blockchain as a Service (BaaS) solutions for enterprises. It offers various features such as pre-built networks, easy deployment of blockchain nodes, built-in tools for smart contract development, and integration with cloud services.</p> <p>Examples of blockchain protocols [37] provided includes:</p> <table border="1" data-bbox="562 810 1912 979"> <tbody> <tr> <td>1. Ethereum</td> <td>2. Avalanche Network</td> <td>3. Hyperledger Besu</td> </tr> <tr> <td>4. Polygon PoS</td> <td>5. Avalanche Subnets</td> <td>6. Quorum</td> </tr> <tr> <td>7. Polygon Supernets</td> <td>8. Hyperledger Fabric</td> <td>9. Corda</td> </tr> </tbody> </table> <p>Examples of products offered by Kaleido platform:</p> <table border="1" data-bbox="562 1070 1912 1182"> <tbody> <tr> <td>1. Blockchain as a Service</td> <td>2. NFT Platform</td> <td>3. CBDC Sandbox</td> </tr> <tr> <td>4. Digital Asset Platform</td> <td>5. Consortium</td> <td>6. Hyperledger FireFly</td> </tr> </tbody> </table>	1. Ethereum	2. Avalanche Network	3. Hyperledger Besu	4. Polygon PoS	5. Avalanche Subnets	6. Quorum	7. Polygon Supernets	8. Hyperledger Fabric	9. Corda	1. Blockchain as a Service	2. NFT Platform	3. CBDC Sandbox	4. Digital Asset Platform	5. Consortium	6. Hyperledger FireFly
1. Ethereum	2. Avalanche Network	3. Hyperledger Besu														
4. Polygon PoS	5. Avalanche Subnets	6. Quorum														
7. Polygon Supernets	8. Hyperledger Fabric	9. Corda														
1. Blockchain as a Service	2. NFT Platform	3. CBDC Sandbox														
4. Digital Asset Platform	5. Consortium	6. Hyperledger FireFly														

2) Hyperledger Fabric

Hyperledger Fabric is a powerful enterprise-grade blockchain architecture known for its flexibility and scalability. It provides a basic platform for developing decentralised apps (dApps) and managing digital assets in enterprise environments. Fabric offers a comprehensive suite of tools and components designed to streamline the development, administration, and interoperability of DLT. Its modular architecture and permissioned nature make it an ideal choice for businesses seeking secure and efficient blockchain solutions tailored to their specific needs. [35]

3) Node.js

The server-side, open-source JavaScript runtime environment Node.js permits the execution of JavaScript code in a browser-independent environment. In other words, allows cross-platform tool for creating scalable network applications. It is commonly used for building scalable network applications and server-side scripting. It ensures smooth data communication between the app and backend services [34].

4) Go (Golang)

Go, alternatively referred to as Golang, is a procedural compiled open-source programming language that has been specifically engineered to facilitate the development of streamlined software applications and the resolution of complex challenges [38]. It is often used for developing backend services, including blockchain applications.

5) Java

Java is a platform-independent, object-oriented programming language that is extensively utilised [39]. It is renowned for its resilience. For the development of enterprise applications, including blockchain solutions, it is frequently employed on account of its robust ecosystem support and scalability.

6) JavaScript	JavaScript is a high-level, interpreted programming language that is a fundamental technology of the World Wide Web, alongside HTML and CSS. It is widely used for creating interactive and dynamic content on websites. JavaScript enables developers to build complex web applications and user interfaces with features that enhance user experience and interaction. [60]
7) Docker Desktop	Docker Desktop is a comprehensive application that provides an integrated environment for building, running, and managing Docker containers on your local machine. It is available for Windows and macOS, offering a user-friendly interface and a suite of powerful features for developers. [62]
8) Docker Compose	Docker Compose is a tool that simplifies the process of defining and managing multi-container Docker applications. It allows developers to define a multi-container environment using a single configuration file, and then use simple commands to manage the deployment, scaling, and operation of these containers. [61]
9) Windows Subsystem for Linux (WSL)	Windows Subsystem for Linux (WSL) is a compatibility layer that enables developers to run a Linux distribution directly on Windows without the need for a virtual machine or dual-boot setup. WSL provides a seamless integration of Linux tools and applications into the Windows environment, facilitating development and testing across both operating systems. [63]

Table 3.2 Software/Tools Required

Chapter 4

System Design

4.1 System Design Diagram

4.1.1 Overview of Hyperledger Fabric

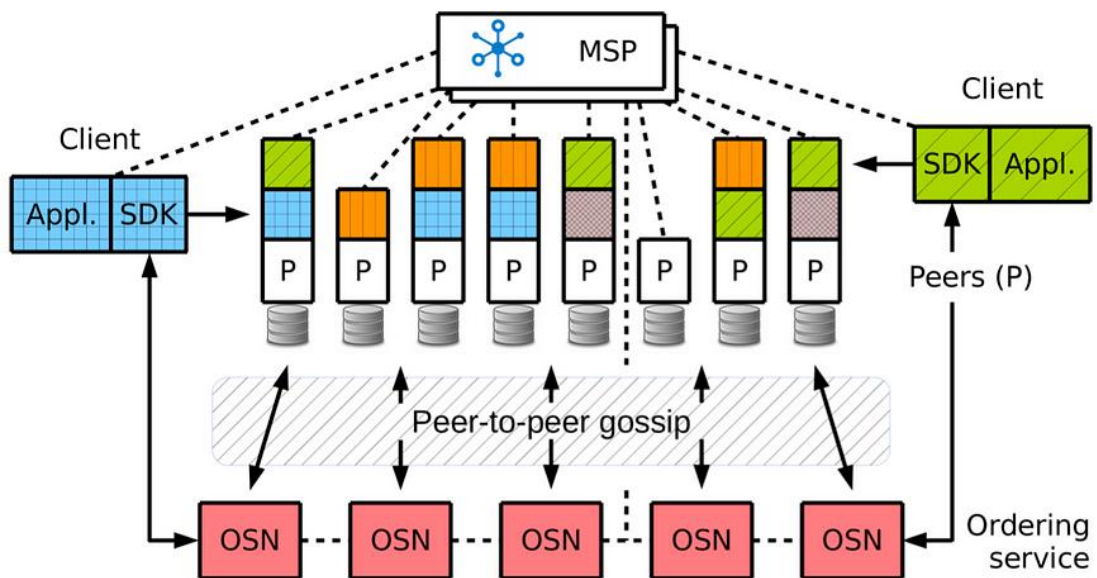


Figure 4.1.1.1 A Fabric network with federated MSPs and running multiple (differently shaded and colored) chaincodes

Figure 4.1.1.1 shows the overall design philosophy of the fabric network. As mentioned in previous literature review, Hyperledger Fabric runs under the architecture of “execute-order-validate” model, and the chaincodes will be passed to nodes with different roles.

Fabric is composed of three main roles: Client, Peer, and Ordering Service. Unlike common node that exists within the same channel and shares the same state, only the designated nodes have the right to access the state of the ledger.

Each Fabric node has the ability to establish several channels concurrently, with each channel functioning as an autonomous blockchain network that maintains its own ledger.

a. Client

The client is the topmost level of the program and acts as the primary interface for users to engage with Fabric using the software development kit (SDK) offered by Fabric.

b. Peer

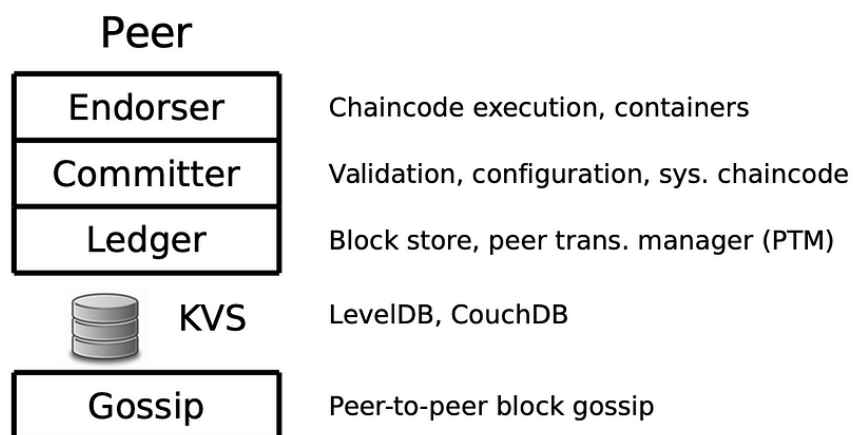


Figure 4.1.1.2 Components of Fabric Peer

The peer plays a crucial role in the Fabric channel. In addition to maintaining blockchain data and ledger state, it is accountable for the "execution" and "validation" of transactions. Based on the "execute-order-validate" model, a limited number of specific nodes are responsible for executing transactions, while the remaining nodes emphasize validation. Meanwhile, the node responsible for executing is referred to as the "Endorsing Peer", whereas the node responsible for validating is known as the "Committing Peer". All peer nodes have the same structure, with each task being executed by its respective module.

Explanations on important modules within the peer component:

- **Endorser:** Implement the "execution" of the transaction. Docker containers are utilized to support chaincode written in various general programming

languages. The client is responsible for deploying the endorsement policy and chaincode in advance.

- **Committer:** Execute the "validation" and state update procedures for the transaction. In addition, configuration files and system chaincode are included.
- **MSP:** MSP is a essential in establishing "permissioned" networks; it is in charge of the node's identity and permission management. Fabric's identification system operates on X.509 certificates, a standard that is well-suited for integration with established identity management systems and public-key infrastructure (PKI). The MSP, on the other hand, is solely tasked with management and not certificate issuance. The Certificate Authority (CA) performs the authority to issue the certificate, whether it is a commercial CA or Fabric-CA, which is native to Fabric.

c. Ordering Service

The ordering service sets Fabric against others. It ensures a systematic arrangement of the transactions and generates blocks for every channel. This service can either function as a single node or as a cluster of multiple nodes. In contrast to the stateful "peer" mentioned earlier, the ordering service is stateless as it does not need to "execute" or "validate". This feature brings a significant advantage to Fabric. By eliminating the need for state synchronization, the ordering service can scale-out more easily.

4.2 Transaction Workflow of Hyperledger Fabric

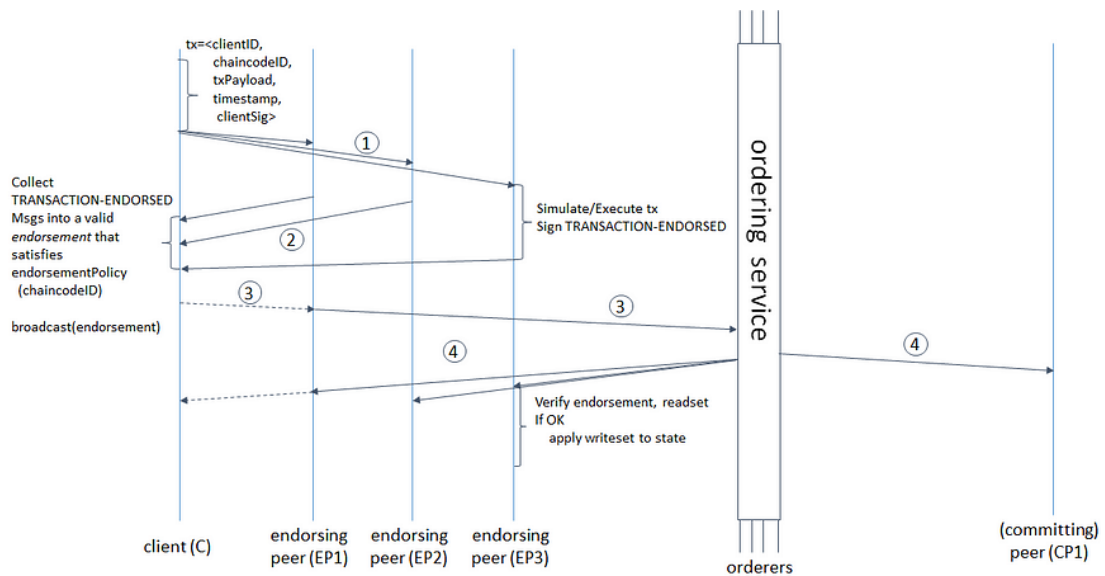


Figure 4.2.1 Transaction Workflow of Fabric

According to the "execute-order-validate" model, the transaction will pass through three stages: "execution", "ordering", and "validation". The sequence diagram above illustrates the overall workflow. Based on the labels in the figure, step 1 and step 2 are associated with "execution," while step 3 and step 4 are associated with "ordering." The paragraph in the bottom text corresponds to "validation."

4.2.1 Execute

Initially, the client submits a transaction proposal along with a digital signature to one or more endorsing peers. The transaction proposal encompasses various elements such as *ChaincodeID* and parameters. It is of utmost importance to include the endorsing policy during the deployment of chaincode, for this policy determines the endorsing peers and the consensus threshold required for execution. For instance, a consensus may require that 3 out of 5 endorsing peers have the same execution result.

Next, the endorsing peers begin executing the transaction to simulate the results. The chaincode will be deployed on the peers beforehand and executed within the Docker container. A versioned key-value pair represents the state of a node. The state that must be read before each execution is referred to as the "read set," while the state that is altered after execution is known as the "write set." Similarly, they both have different versions.

The certifying peer encapsulates the execution result within a "endorsement" package and transmits it to the client. The content comprises of a read set, write set, and an assortment of meta data including *TransactionID*, *EndorserID*, and the signature of the endorsing peer. Meanwhile, the client gathers "endorsements" from various endorsing peers until the endorsement policy's consensus threshold is reached. This means that a specific number of endorsing peers must have the identical execution result, including the same read set and write set.

4.2.2 Order

Once the endorsing policy is met, the client proceeds to encapsulate the execution result within a transaction and forwards it to the ordering service. If the policy is not satisfied, the transaction is simply discarded. The transaction includes the data payload, metadata, and endorsements from every endorsing peer. The ordering service will process all the transactions received within a period of time and generate a block.

Subsequently, all the committing nodes consisting of all transactions will be distributed by the ordering service. In this scenario, the ordering service acts as the central

coordinator, resulting in a more streamlined message propagation compared to the gossip protocol. As a result, the communication complexity is significantly reduced from $O(n^2)$ to $O(n)$. While the bandwidth of the ordering service may pose a limitation, one way to mitigate this is by utilising a larger cluster, which can help minimise communication overhead.

4.2.3 Validate

Ultimately, the committing node will verify the transactions once it receives the block. This process is incredibly efficient and speedy since it requires only the transaction to be executed. Validation can be broken down into three distinct steps: Initially, the evaluation of the endorsing policy involves verifying if the endorsement linked to the transaction aligns with the endorsing policy. Following that, the read/write conflict check guarantees the consistency of the version of the "read set". If no issues arise from the aforementioned checks, the committing peers proceed to update the ledger based on the write set.

It is noteworthy that the committing node must keep track of whether the block contains any invalid transactions. If a transaction is deemed invalid, it is immediately discarded. In Fabric, a distinct countermeasure can be employed when Byzantine faults are detected: Add the endorsing peer to the blocklist.

4.3 Supply Chain Architecture Diagram

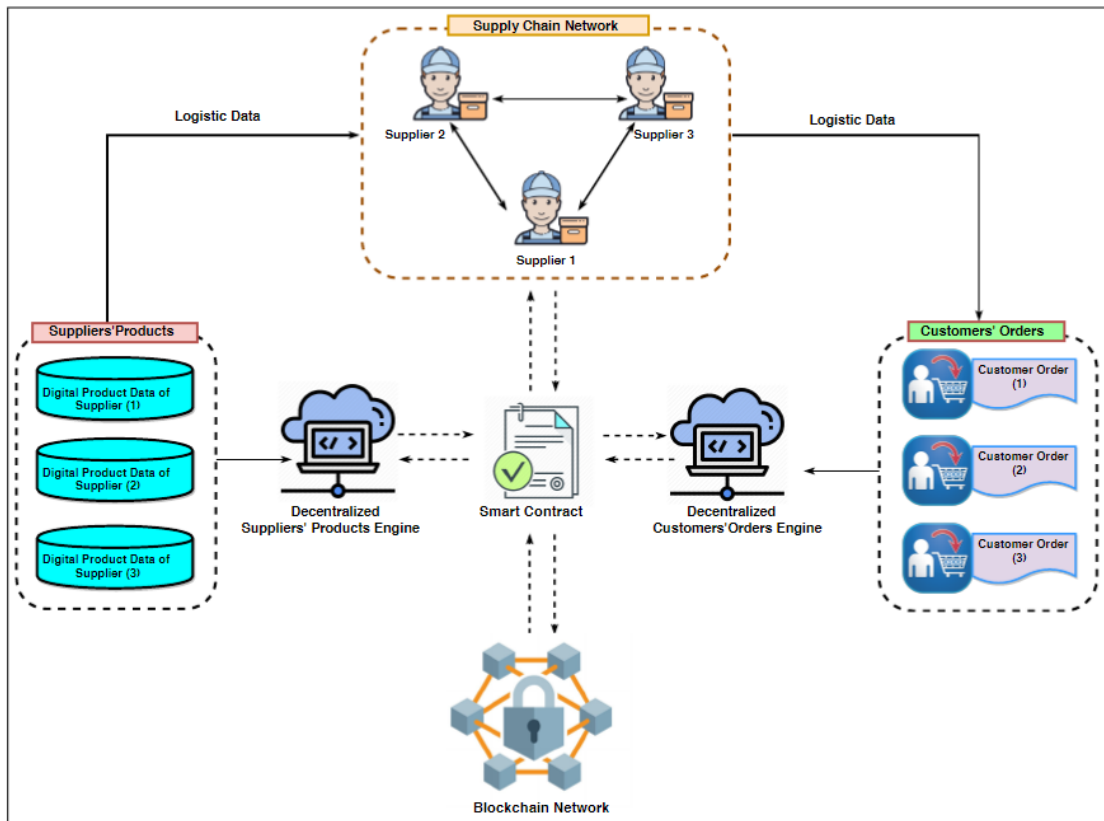


Figure 4.3.1 Managing Supply Chain Process using Blockchain Technology

In this approach, the primary function of blockchain technology is to keep the most recent supply chain transactions involving different consumer needs and various supplier products.

Customers can generate fresh demand by using the decentralized customer orders engine, while suppliers can offer new products using the decentralized supplier products engine.

The smart contract functions as an authentication protocol between the two engines, validating and safeguarding provider-to-client transaction data patterns. Subsequently, the validated transactions are securely documented as a novel block and appended to the blockchain.

4.4 Hyperledger Firefly Development Stack [64]

The Hyperledger Firefly Development Stack is a group of Supernodes built in to cooperate on a single development system. In this development stack, many members, also known as organizations, are created and each of them has a unique Supernode. By virtue of the Supernode, developers can create and test data flows in the development environment, which is made up of the private and public data between different parties.

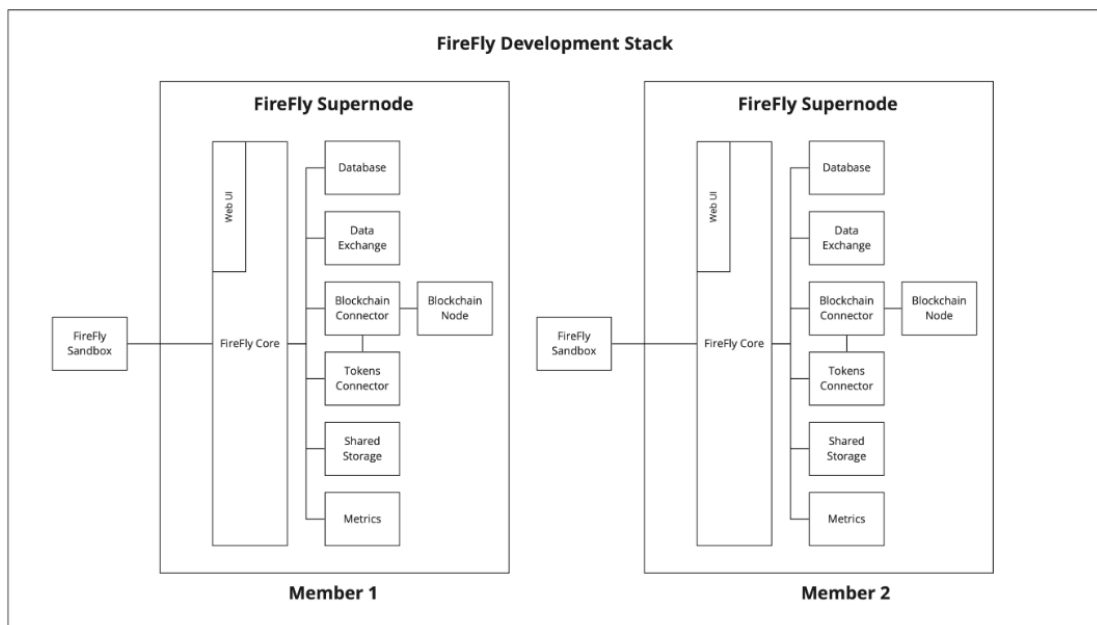


Figure 4.4.1 Firefly Development Stack

The deployment of development stack creates a Firefly Sandbox for every member within the instances. It is known as the end-user application that use the Firefly's API, and users are able to glance through the backend and frontend of the features provided by the Firefly. It also offers code snippets as examples of how to incorporate those features into their own applications.

Chapter 5

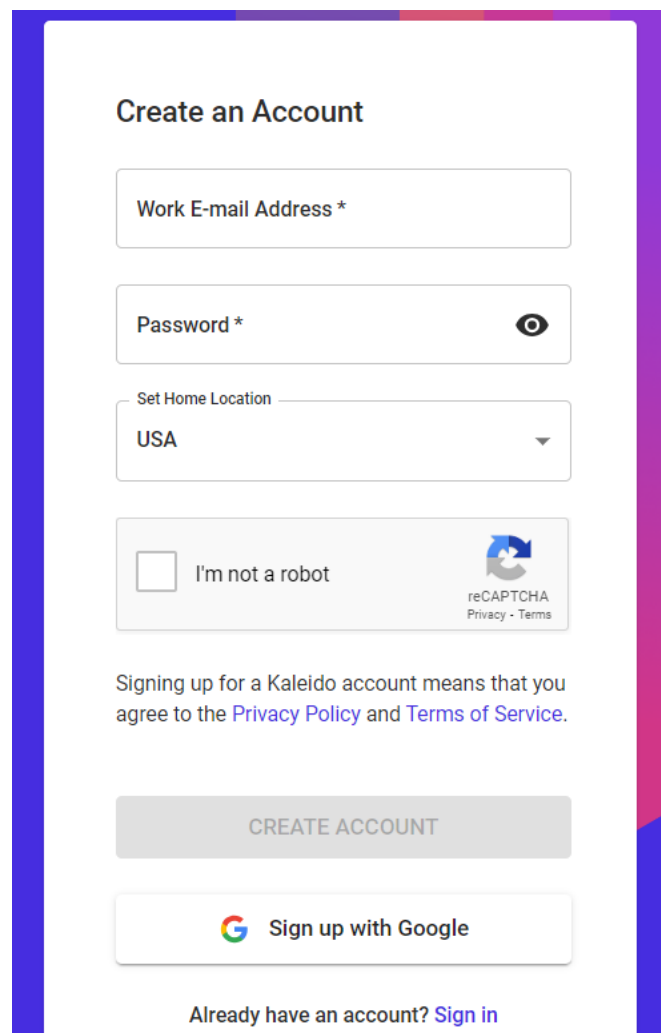
Preliminary Work

5.1 Setting Up

5.1.1 Software

Sign-up Kaleido Account

Before the development of the project, there are setup need to be done. First, we need to sign up for a new account on Kaleido platform.



The image shows a web form titled "Create an Account" for the Kaleido platform. The form is enclosed in a blue and purple border. It contains the following elements:

- A text input field labeled "Work E-mail Address *".
- A password input field labeled "Password *" with an eye icon for toggling visibility.
- A dropdown menu labeled "Set Home Location" with "USA" selected.
- A reCAPTCHA widget with the text "I'm not a robot" and a checkbox.
- A paragraph of text: "Signing up for a Kaleido account means that you agree to the [Privacy Policy](#) and [Terms of Service](#)."
- A grey button labeled "CREATE ACCOUNT".
- A button labeled "Sign up with Google" with the Google logo.
- A link at the bottom: "Already have an account? [Sign in](#)".

Figure 5.1.1.1 Sign Up

5.2 Implementation on Kaleido Platform

Step 1: Create Network

- After signing up, create an environment for the project.
- Name the network as “*MySupplyChainManagement*”
- Click on the “*next*” button.
- Select *Ohio* as home region.

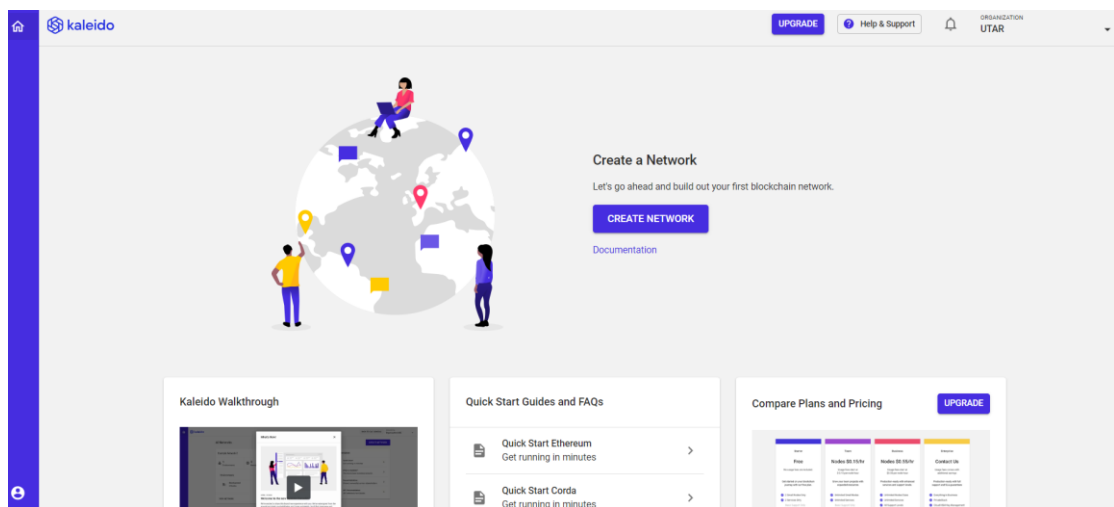


Figure 5.2.1 Main Page of Kaleido

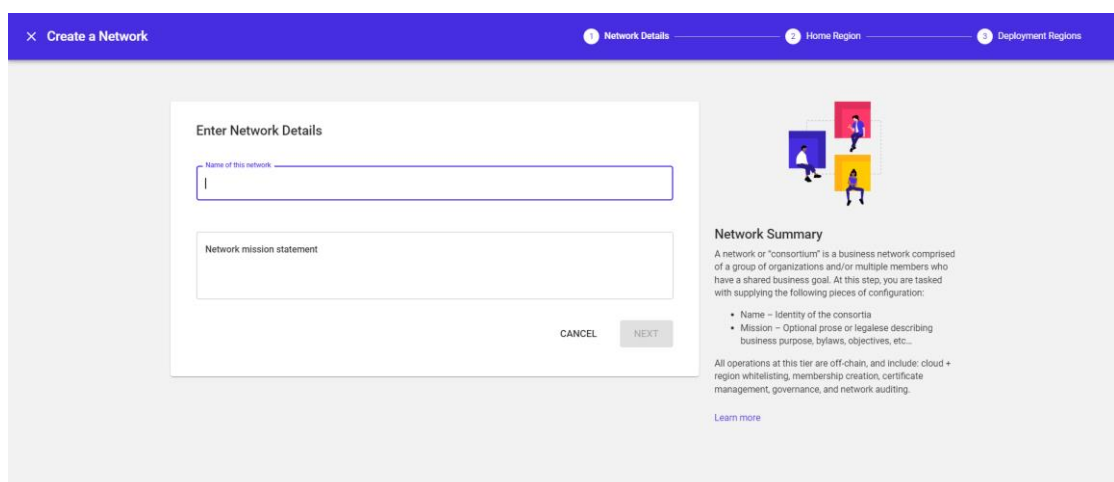


Figure 5.2.2 Create Network

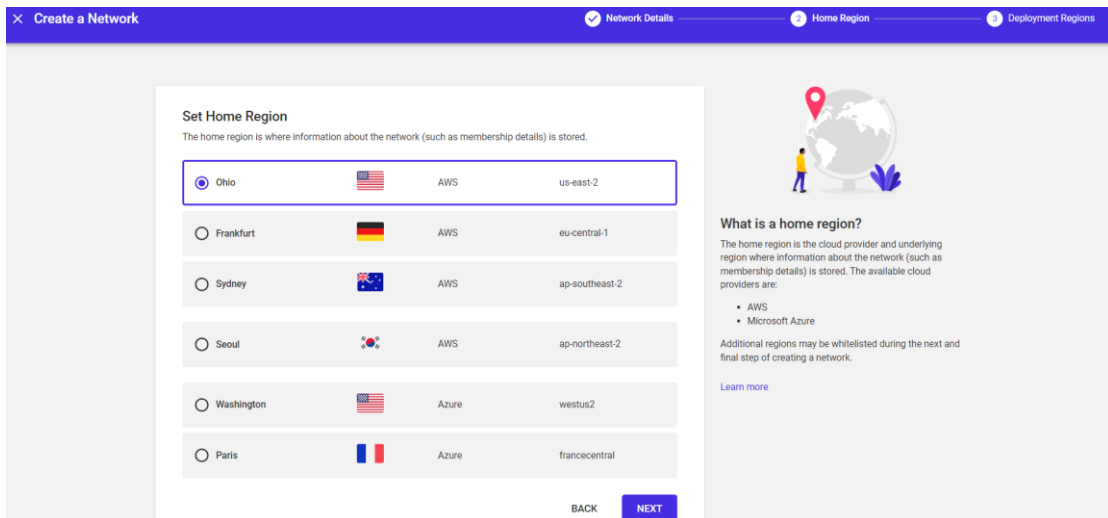


Figure 5.2.3 Home Region

Step 2: Create Environment

- Create a network named *“MySupplyChainManagement”*
- Then, create a sandbox environment (Figure 5.1.6)
- Select the Standard Blockchain Service as the environment type.

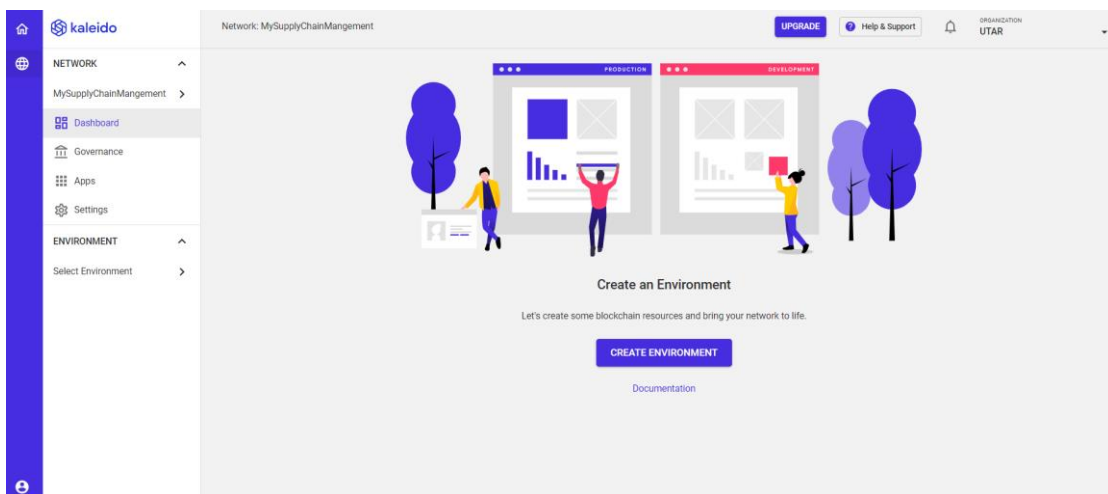


Figure 5.2.4 Kaleido Dashboard

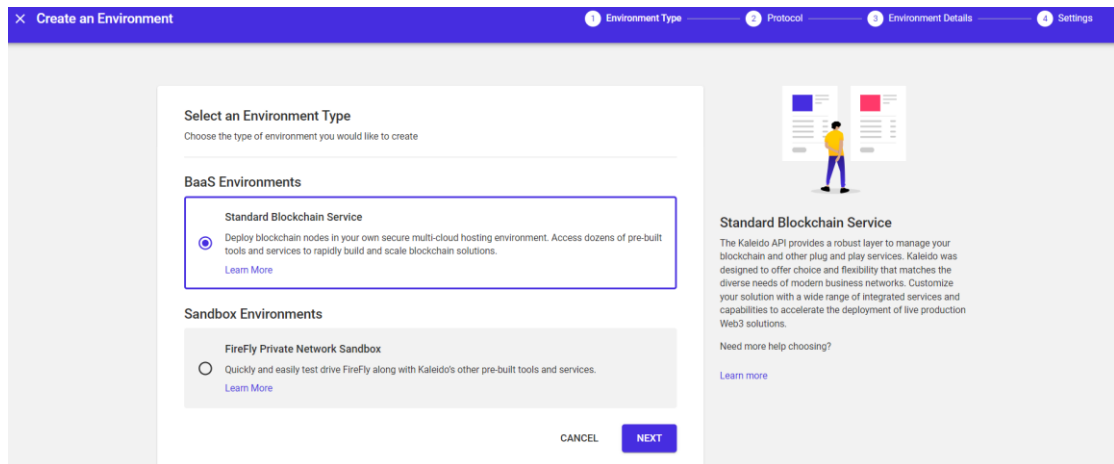


Figure 5.2.5 Environment Type

- Moving on to the next section is to select an environment protocol.
- In this case, select the Hyperledger Fabric as our environment protocol.

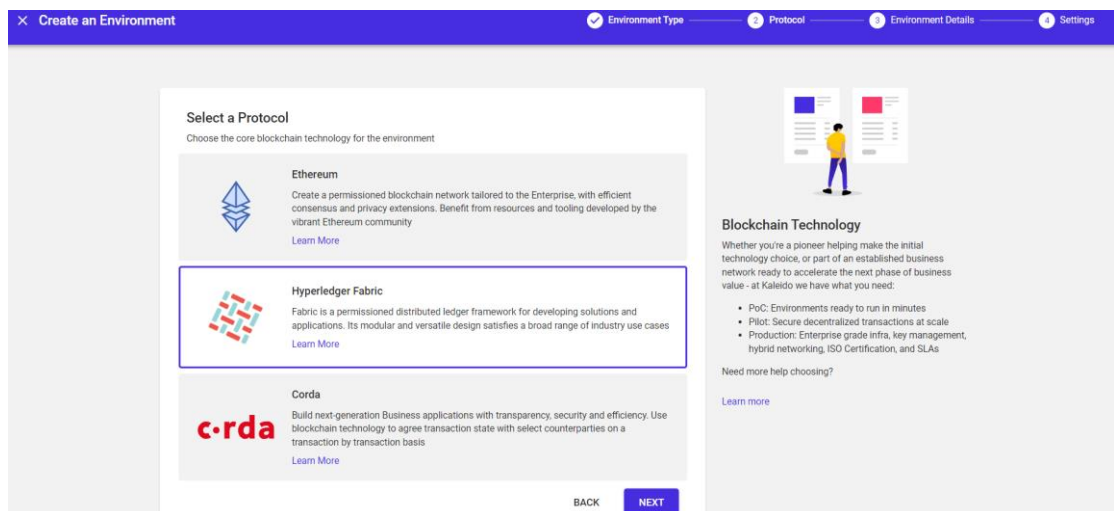


Figure 5.2.6 Environment Protocol

- Next, name the environment as “*MyFYP*”.
- Click on next.
- Finally, select the “*Hyperledger Fabric 2.4*” as the provider, whereas the consensus algorithm as *Raft* and State Database as *LevelDB*.

Figure 5.2.7 Environment Details

Figure 5.2.8 Environment Provider Settings

Step 3: Create Orderer Node, Peer Node and Elect Certificate Authority

- Click on the create node.
- UTAR is created as organisations and named it “*Orderer1*”
- Create a peer node and named it as “*Peer1*” node.
- Electe “*UTAR*” as the CAs.

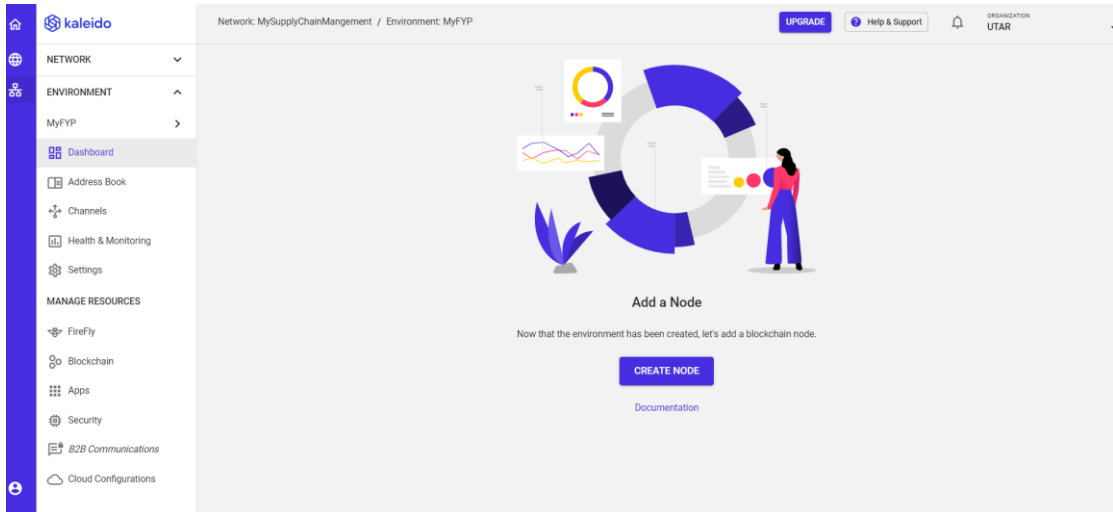


Figure 5.2.9 Environment Dashboard (a)

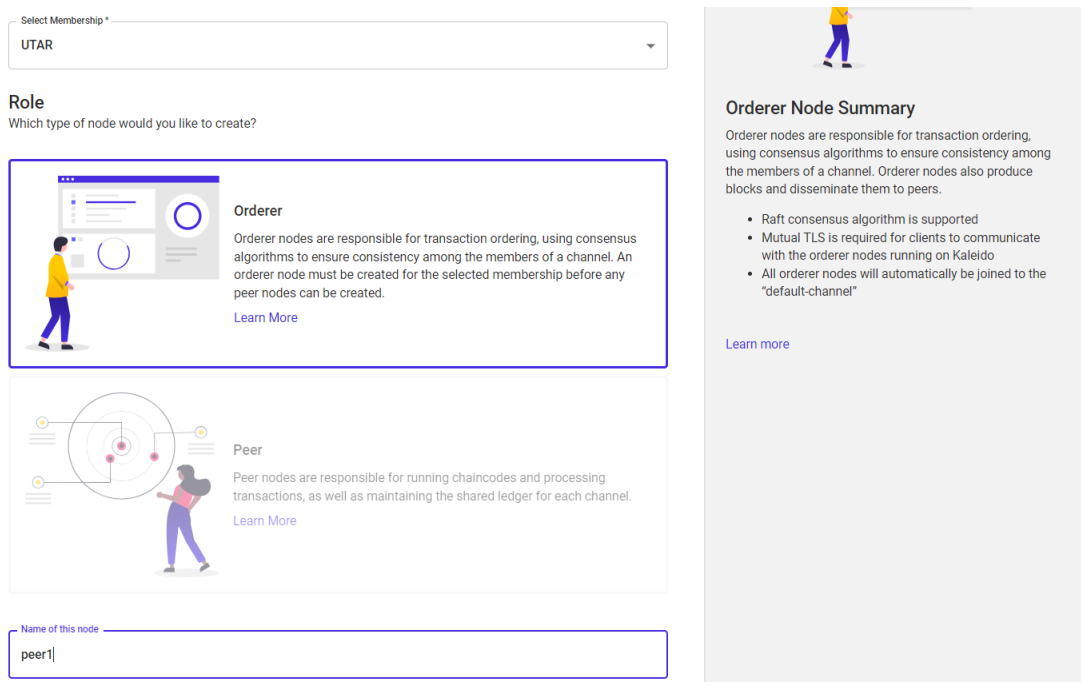


Figure 5.2.10 Creation of Node

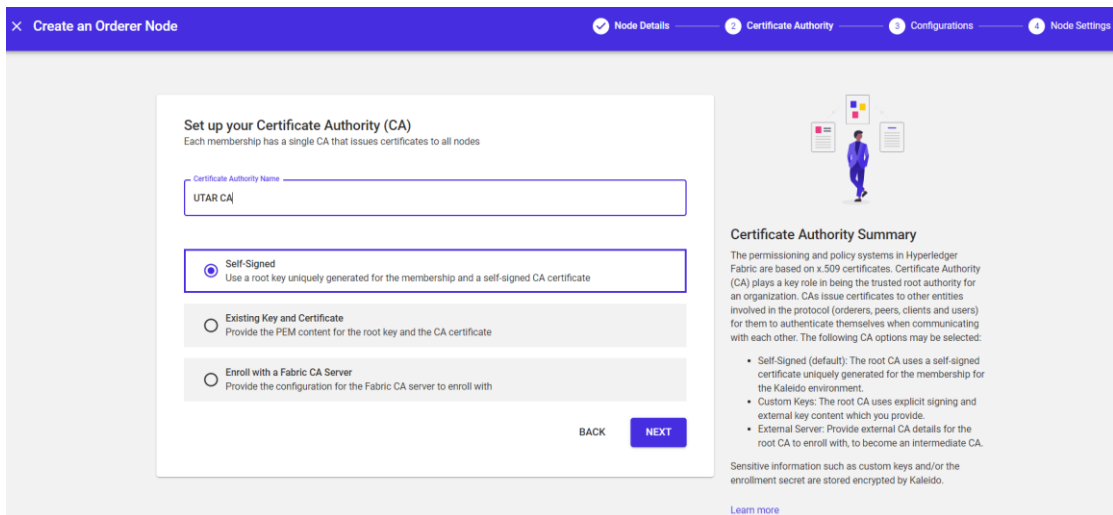


Figure 5.2.11 Setup Certificate Authority (CA)

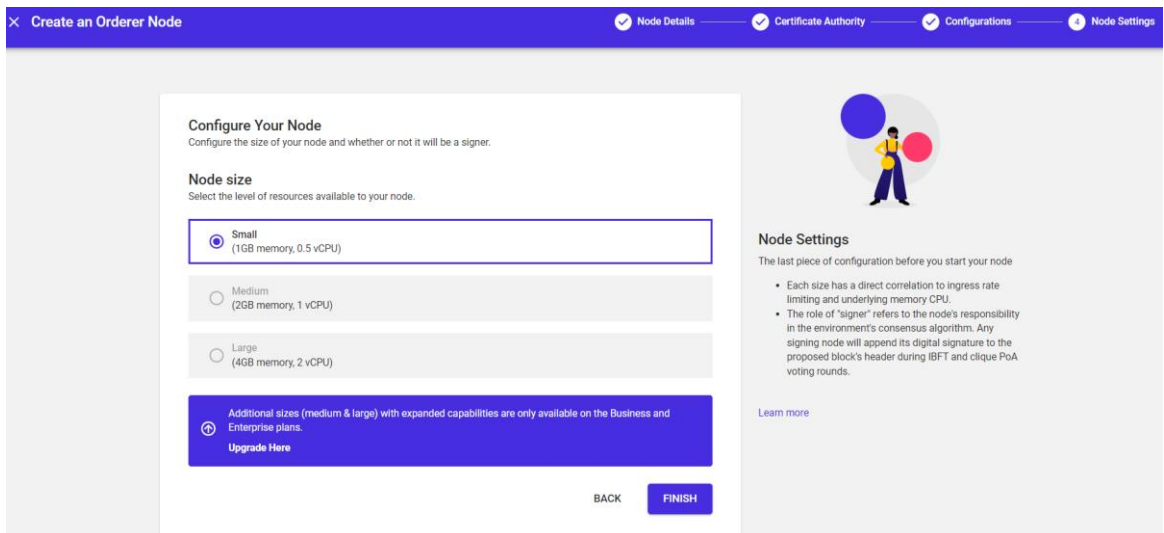


Figure 5.2.12 Node Configuration

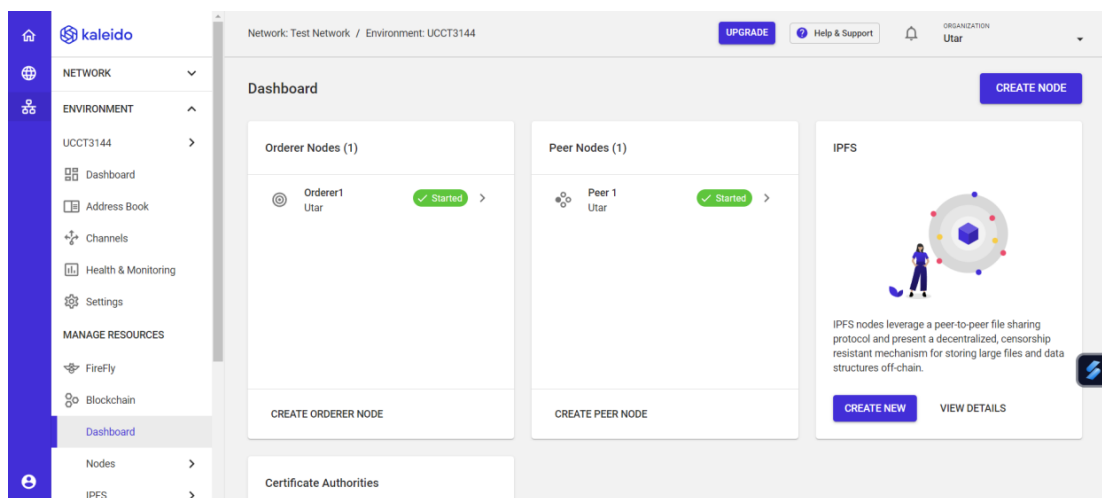


Figure 5.2.13 Environment Dashboard (b)

CHAPTER 5

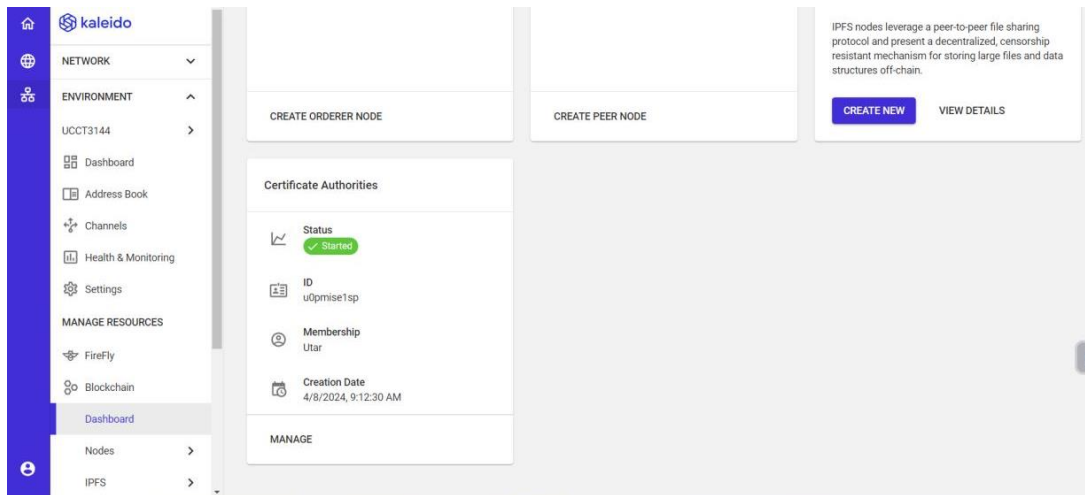


Figure 5.2.14 Environment Dashboard (c)

Step 4: Create IPFS

- Create an InterPlanetary File System (IPFS) and named it as “*IPFS1*”.

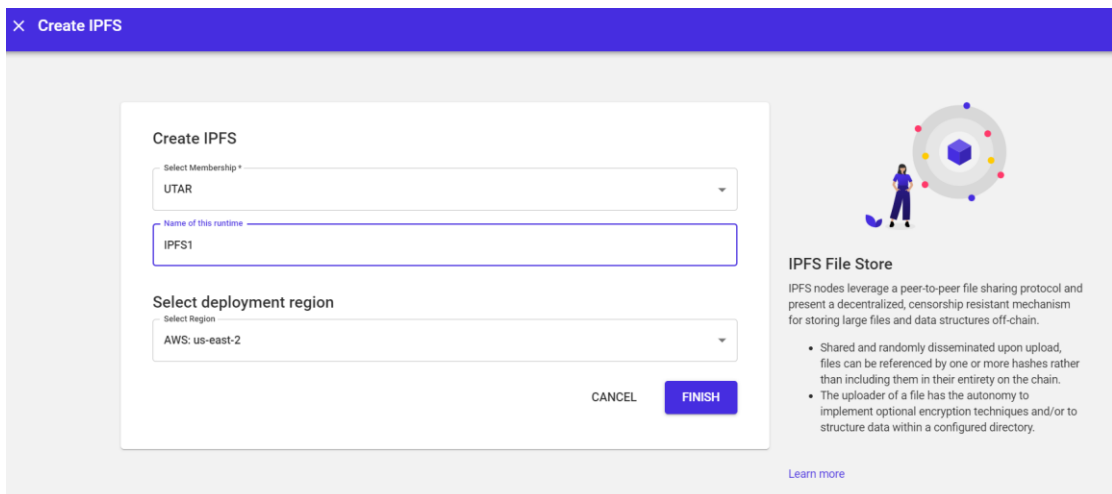


Figure 5.2.15 Create IPFS

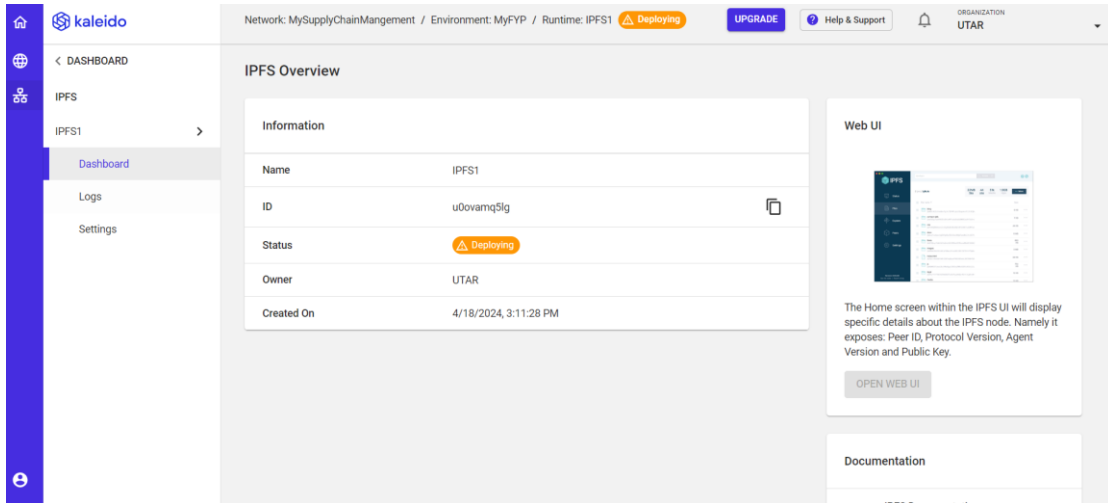


Figure 5.2.16 IPFS Dashboard

Step 5: Create Channel

- Create a new Channel.
- Go to the *Environment > Channel Page* and click on *Add Channel*.
- Name the channel as “*mychannell*”.
- Add preferred members or organisations to join the channel
- Click on next and finish for the policy page.

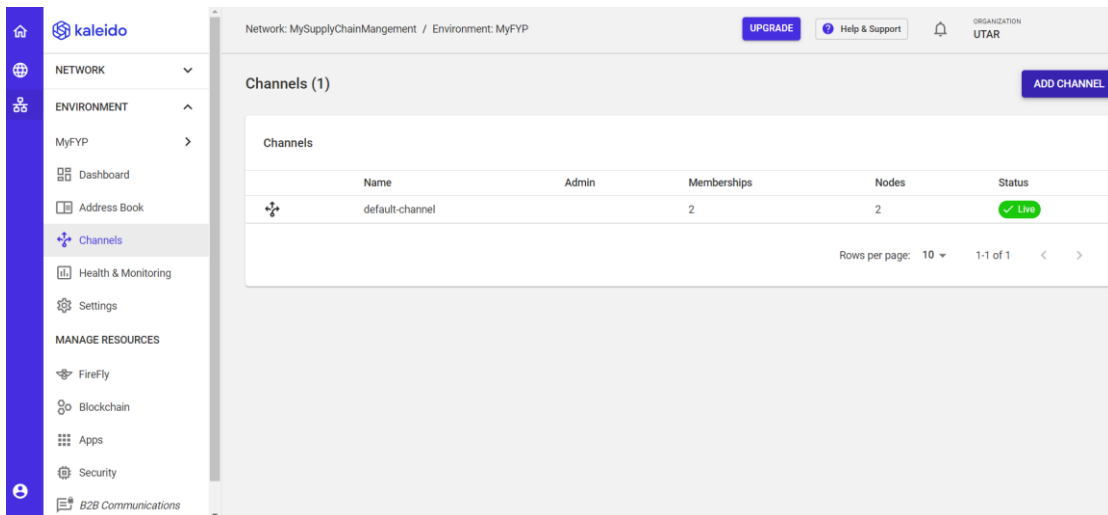


Figure 5.2.17 Channel Page

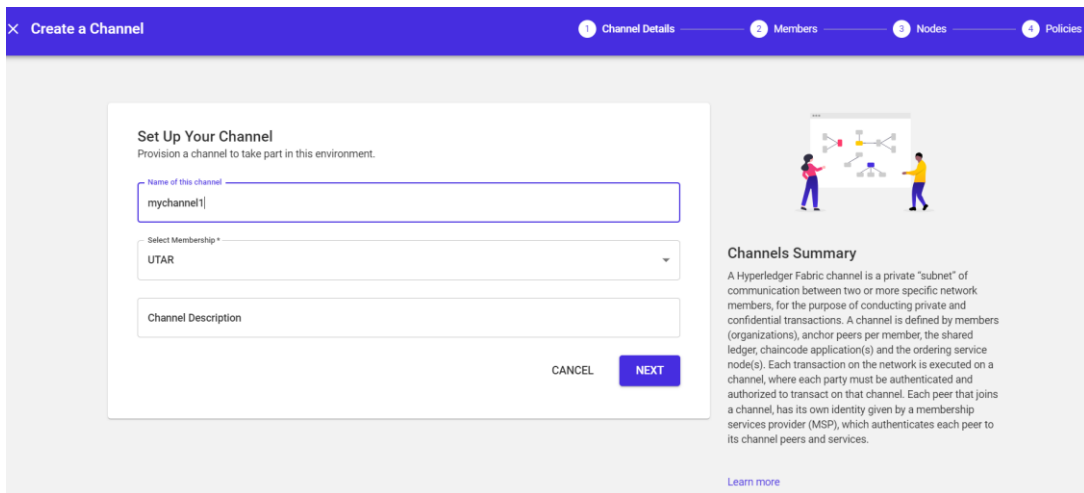


Figure 5.2.18 Create Channel

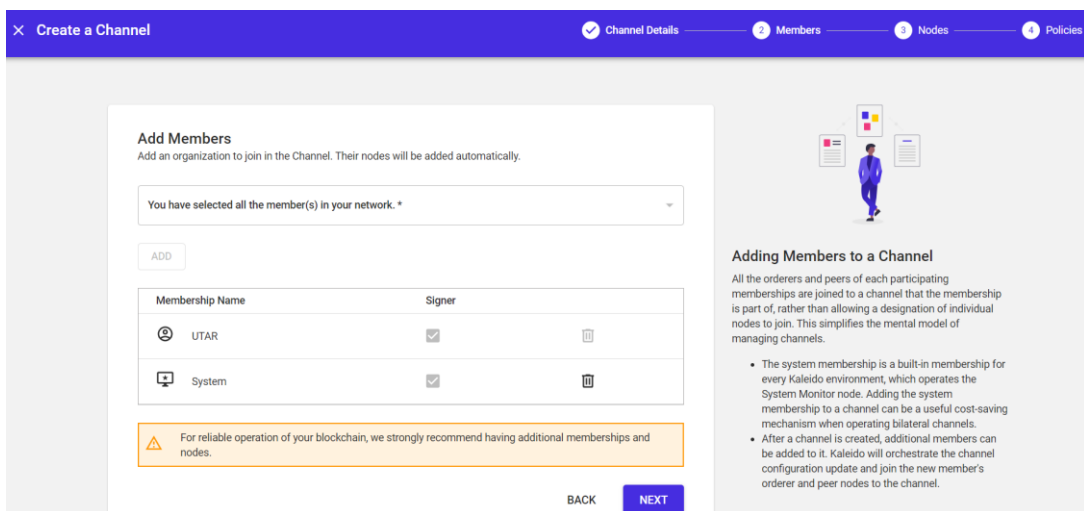


Figure 5.2.19 Add Channel Members

Step 6: Create App

- In *Network > Apps*, click on *create app*.
- Select “*Hyperledger Fabric*” as the protocol in the app creation.
- Name the app as “*Asset Transfer*”.
- The source code used is precompiled *Go language, or Golang*.

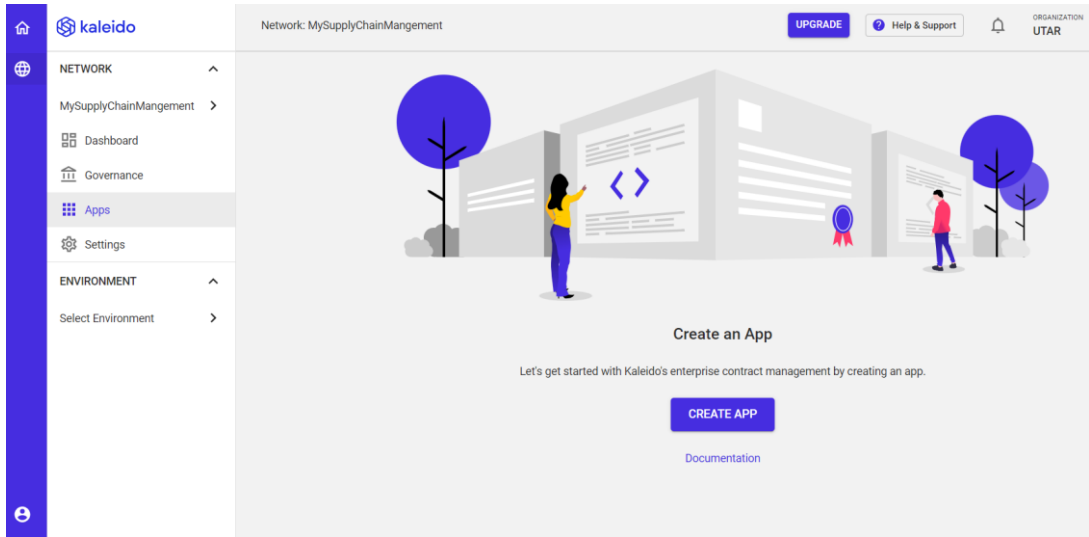


Figure 5.2.20 Apps Page (a)

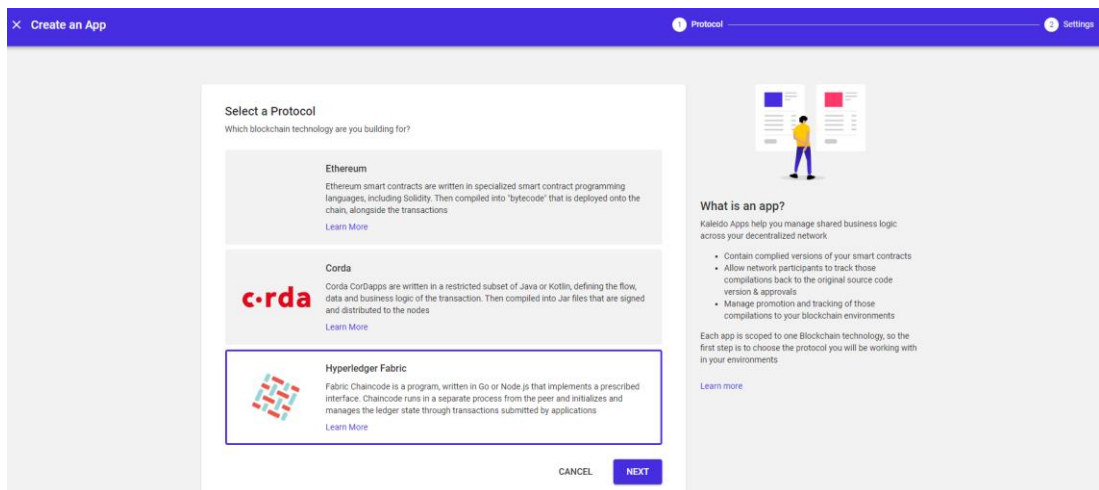


Figure 5.2.21 Create App Page

Enter App Details

Create your app by specifying a name and telling us where the source code for this project is located.

Select Membership *
 UTAR

Name
 Asset Transfer


Description (optional)

Source code
 Kaleido supports uploading a precompiled Golang executable file (.bin), or, a zipped Node.js project directory.

Golang Executable
 Select to upload a precompiled Golang executable file with a .bin extension.

Node.js Project
 Select to upload a zipped Node.js project directory (including Typescript). Kaleido will install your project's node modules during chaincode packaging.

[BACK](#) [FINISH](#)



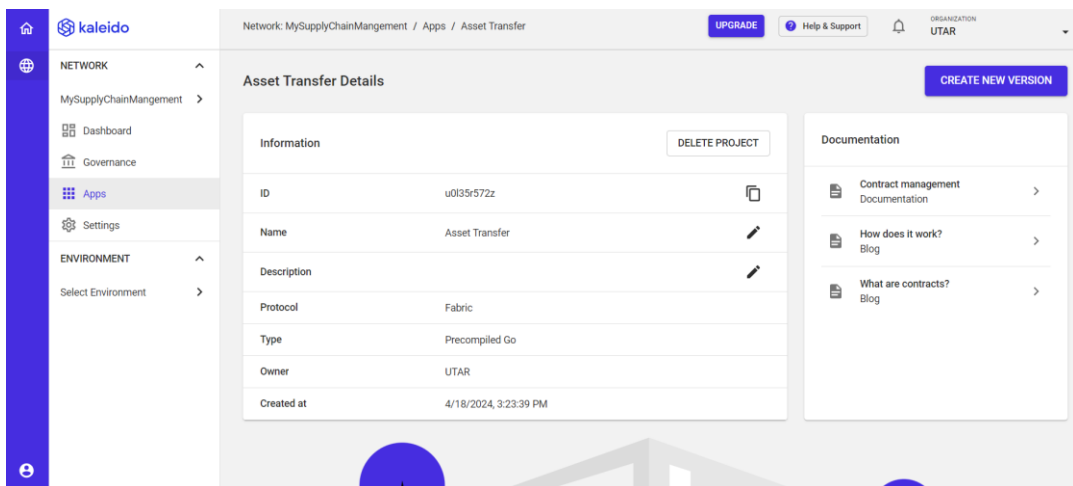
Setting up your App

The project will be visible to all memberships in a network

- Membership - if you manage multiple memberships in the network, pick the one that you are acting as.
- Name and description - Give your project a unique name and an optional description.

[Learn more](#)

Figure 5.2.22 Enter App Details



The screenshot shows the Kaleido web interface. On the left is a navigation sidebar with categories like NETWORK, ENVIRONMENT, and Apps. The main content area is titled 'Asset Transfer Details' and contains a table with the following information:

Information		DELETE PROJECT
ID	u0l35:572z	
Name	Asset Transfer	
Description		
Protocol	Fabric	
Type	Precompiled Go	
Owner	UTAR	
Created at	4/18/2024, 3:23:39 PM	

On the right side of the details panel, there is a 'Documentation' section with links to 'Contract management Documentation', 'How does it work? Blog', and 'What are contracts? Blog'. A 'CREATE NEW VERSION' button is located at the top right of the details panel.

Figure 5.2.23 App Page (b) - Asset Transfer Created

Step 7: Create Configuration File, Create and Deploy Chaincode

- Click on the *create new version* on the top right corner
- Enter the chaincode details to create a chaincode.
- A bin file is being created by running the code `go build -o assetTransferV1.bin`.
- To compile the binary file, installed GO package, download Hyperledger Fabric Samples from GitHub.
- Then run the code highlighted above after directing to the Hyperledger Fabric Samples on command prompt.
- Do not click on the *require initialization before invocation* option.

Enter Chaincode Details
Select the Membership who will own this version and give this specific version a description.

Select Membership *
UTAR

Description *
1.0.0

Select a file
Select the file to upload

assetTransferV1.bin **SELECT FILE**

Require initialization before invocation
This adds the "--init-required" parameter to the lifecycle commands, which requires the chaincode to be initialized before it can process transactions.

FINISH

Your chaincode version
Set the details for

- Owner - the membership you are acting under when uploading this subscription.
- Description - Give this version a description so you will recognize it later.
- Binary file - upload the linux compiled go binary for distribution to nodes in the network

[Learn more](#)

Figure 5.2.24 Enter Chaincode Details

- It then shows the new version within the app project namespace.
- Next, click on the promote to environment to promote it to the existing environment.
- Select the environment as *"MyFYP"*.

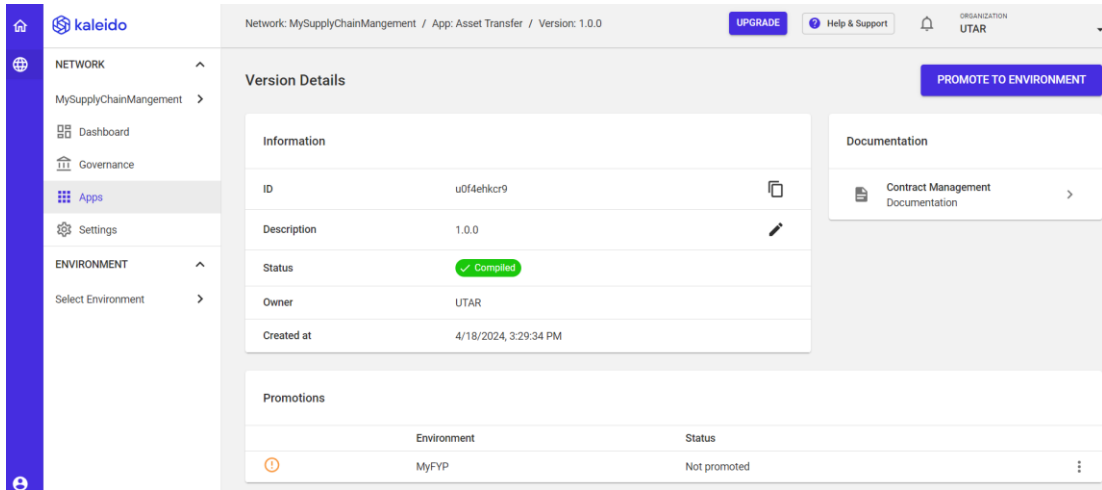


Figure 5.2.25 App: Asset Transfer Version Details

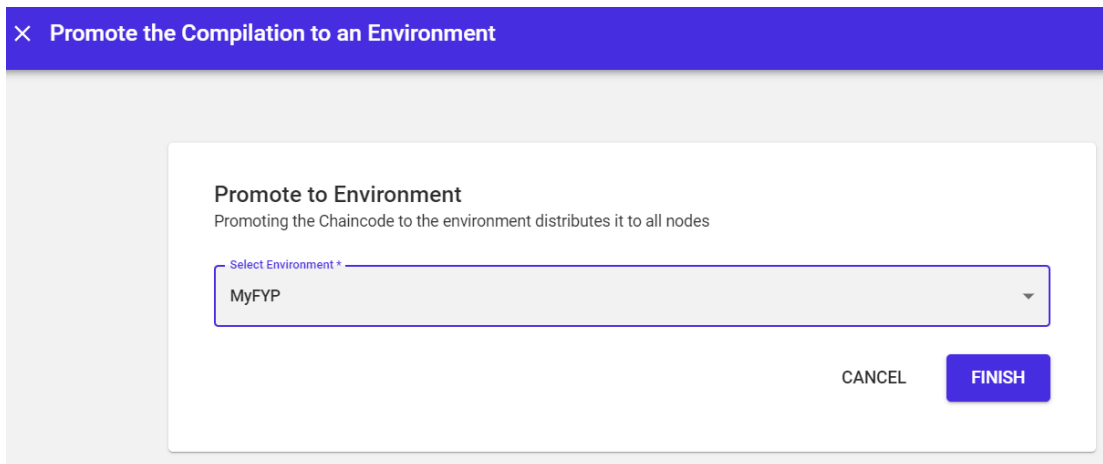


Figure 5.2.26 Promote to Environment

- After a promotion we are taken to the "Chaincodes" tab within the environment.
- Click the *deploy instance to channel* button in the top right of this page to deploy.
- Select the channel as “*mychannell*”.

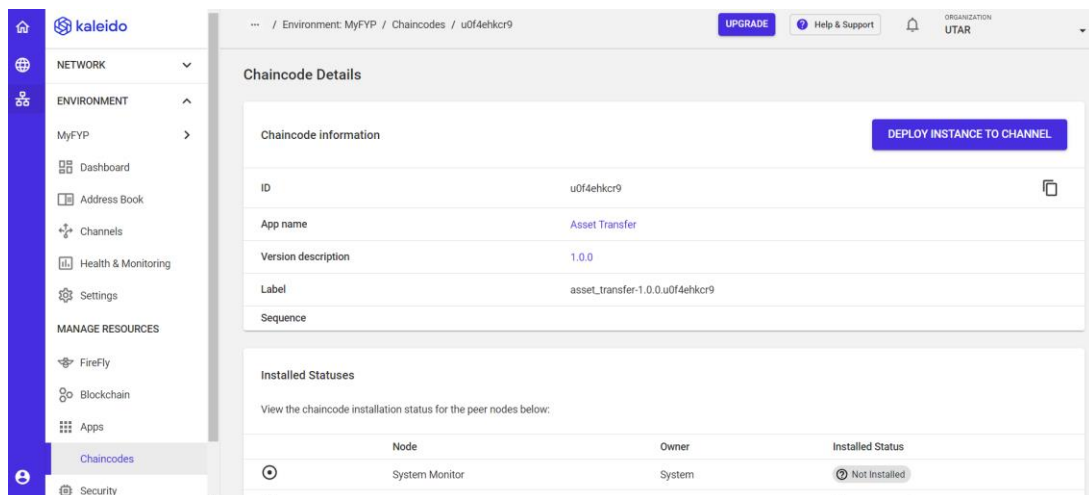


Figure 5.2.27 Deploy Instance to Channel

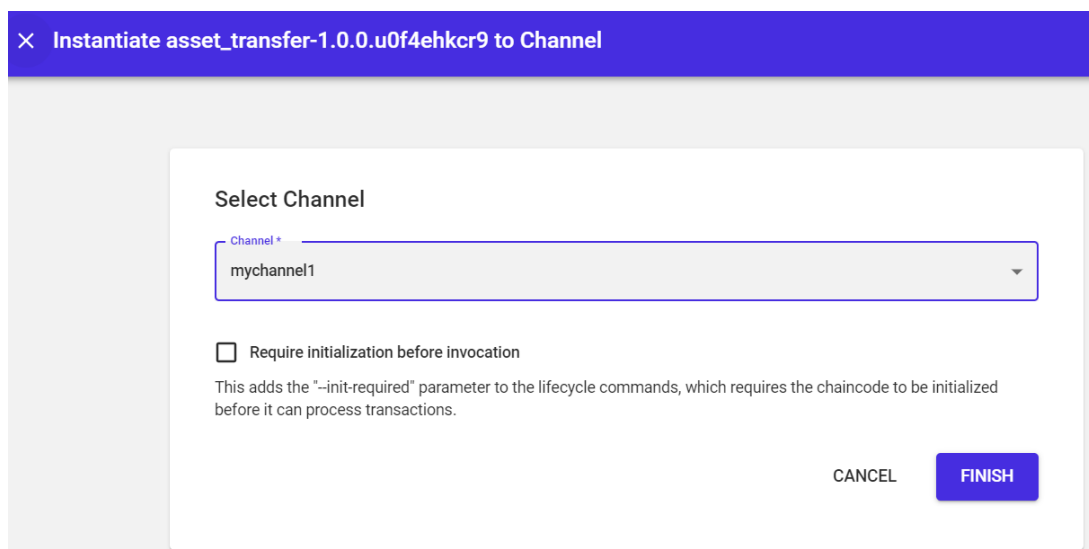


Figure 5.2.28 Select Channel

- After clicking the *finish* button, it will be directed to the chaincode page.
- Scroll to the bottom of the screen, and the deployed channel will be shown.

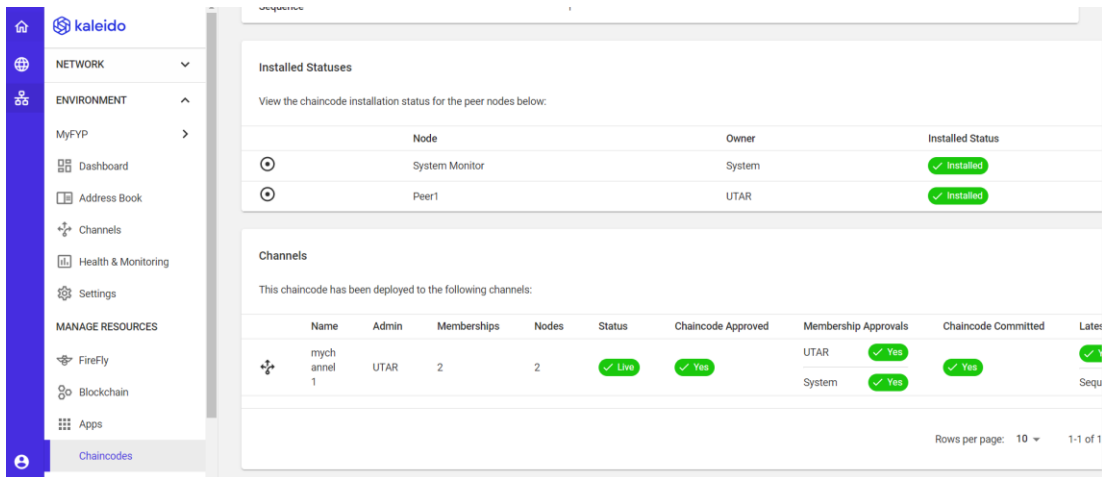


Figure 5.2.29 Chaincode Page

- To ensure the chaincode is running fine, navigate to the node page of *mychannell* through *Environment > Manage Resources > Dashboard > Peer Nodes*.
- Click on *Peer1* to direct to the peer node and visit the log activity by clicking the “Logs” on the left side of the navigation bar.
-

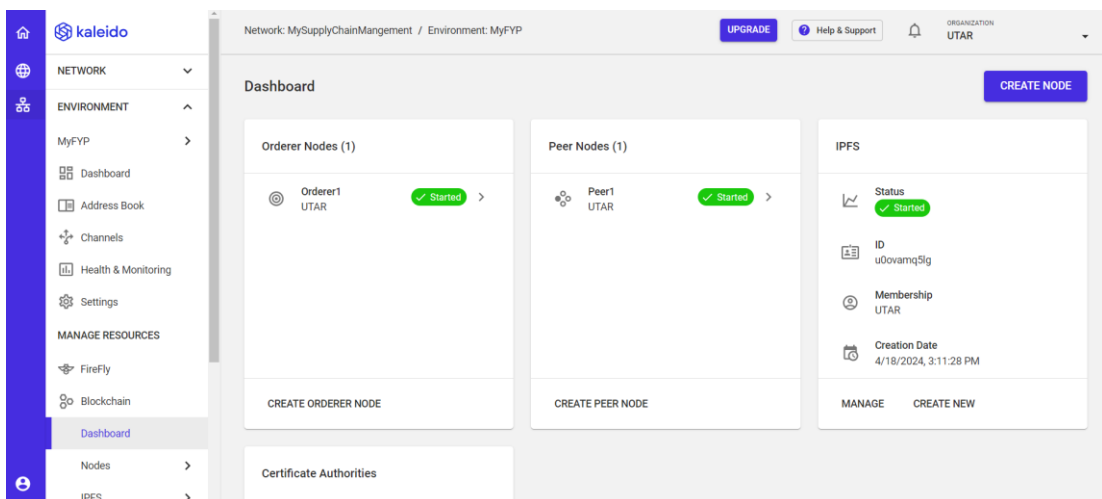


Figure 5.2.30 Blockchain Dashboard

- The successful instantiation is confirmed by referring to the log. It acknowledges us know that we can proceed to invoke the chaincode and send transactions into the chain.

```
Successfully queried chaincode name 'asset_transfer' with definition
{sequence: 1, endorsement info: (version: '1.0.0.u0f4ehkcr9', plugin:
'esc', init required: false), validation info: (plugin: 'vsc',
policy:
'12202f4368616e6e656c2f4170706c6963617469666e2f456e646f7273656d656e74
'), collections: {}},
```

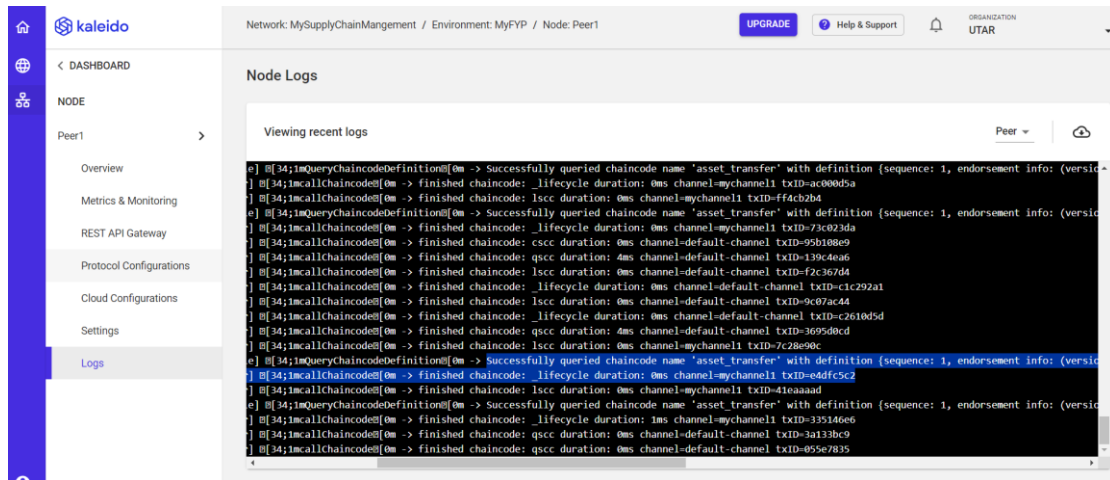


Figure 5.231 Peer1 Logs

Step 8: Run Application

- In the peer node page, click the Open Web UI button.

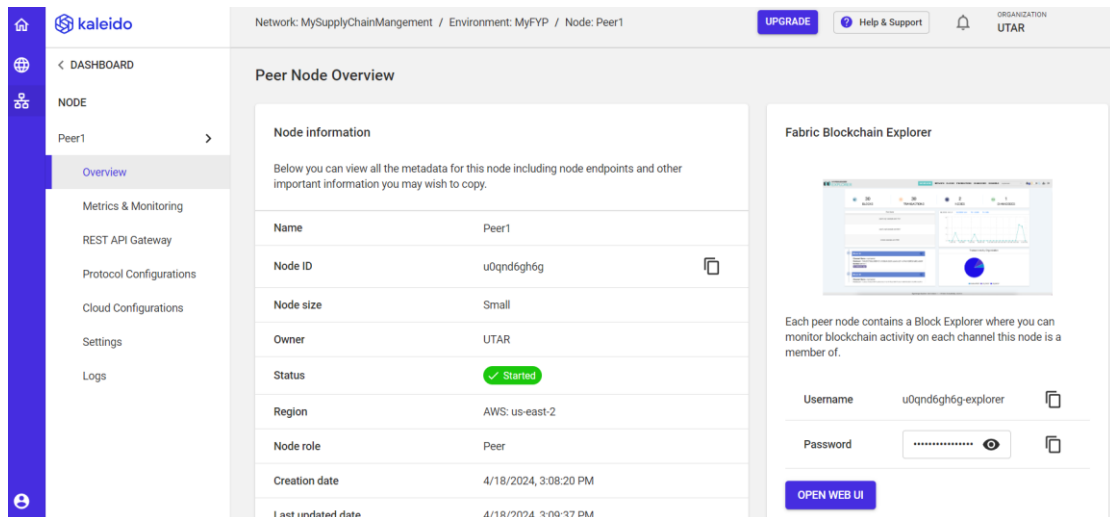


Figure 5.232 Peer Node Overview

CHAPTER 5

- It will direct users to the sign-in webpage, and users are to login to the explorer using the username and password given on the peer node page.
- After signing in, it directs users to the dashboard of Fabric Blockchain Explorer.

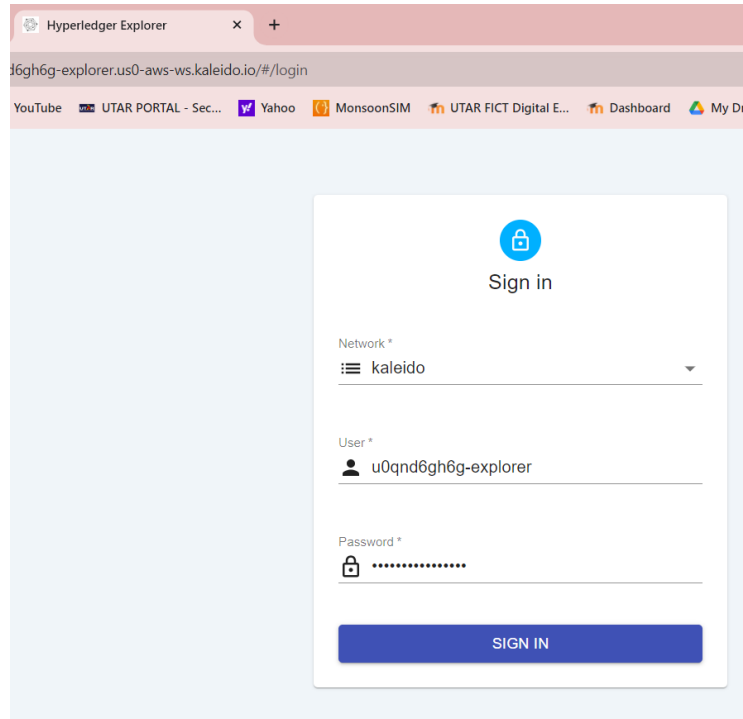


Figure 5.2.33 Fabric Blockchain Explorer - Sign in Page

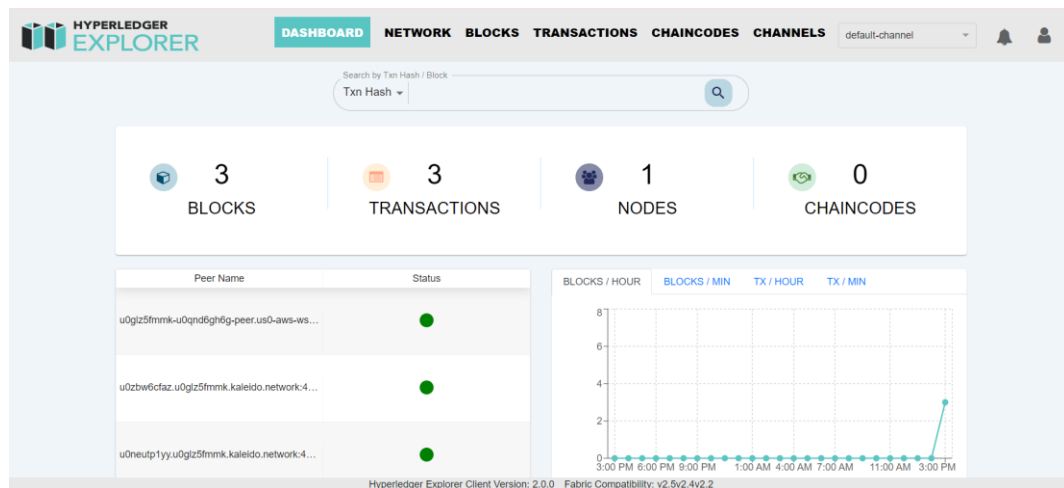


Figure 5.2.34 Fabric Blockchain Explorer Webpage Dashboard

- Click on the “Channels” button on the top navigation, and it shows the transaction records in Transactions and Ledger records in Blocks, Available peers and orderers in network, and available channels.

ID	Channel Name	Blocks	Transactions	Timestamp
3	default-channel	3	3	2024-04-18T07:04:14.000Z
4	mychannel1	6	6	2024-04-18T07:17:02.000Z

Figure 5.2.35 Fabric Blockchain Explorer - Channels Page

Step 9: Register, Enrol Signing Account

- Returning the Peer1 Node page, click on the visit Rest API button.

The screenshot shows the Kaleido interface for a Peer1 node. On the left is a navigation menu with 'Peer1' selected. The main content area displays node metadata and a REST API Gateway section. The REST API Gateway section includes a 'VIEW REST API' button and a description of Hyperledger FireFly FabConnect. To the right, there is a 'Connect your node' section with a password field, an 'OPEN WEB UI' button, and a list of endpoints for Peer, Blockchain Explorer, and REST API Gateway, each with a 'CONNECT APP' button.

Figure 5.2.36 Node Page – View Rest API

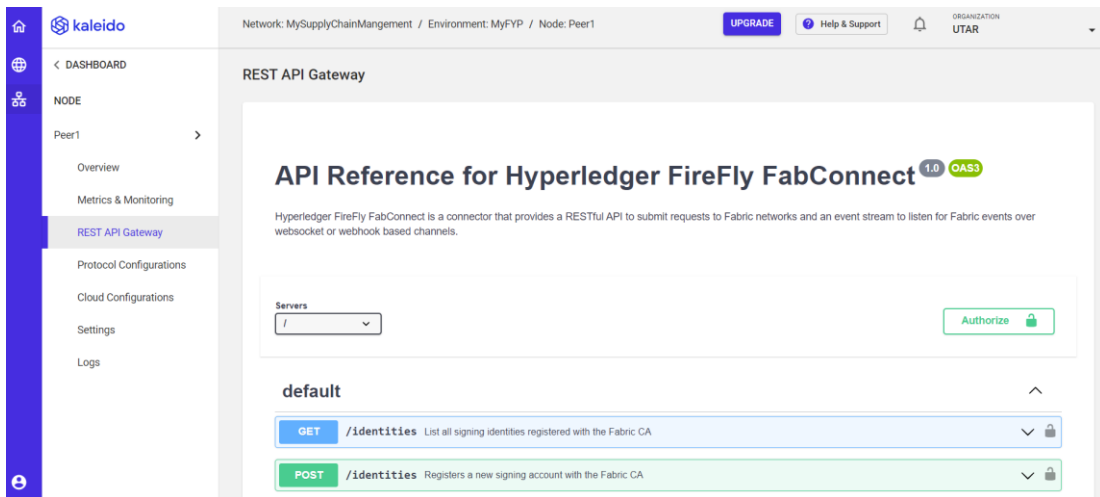


Figure 5.2.37 Rest API Gateway

Register a User

- First, register a user with Certificate Authority.
- **POST** to the **/identities** route with name and type arguments.
- The *maxEnrollments* and *attributes* arguments are optional, and we will forgo them.
- To edit and execute the API, click on “Try it out”.

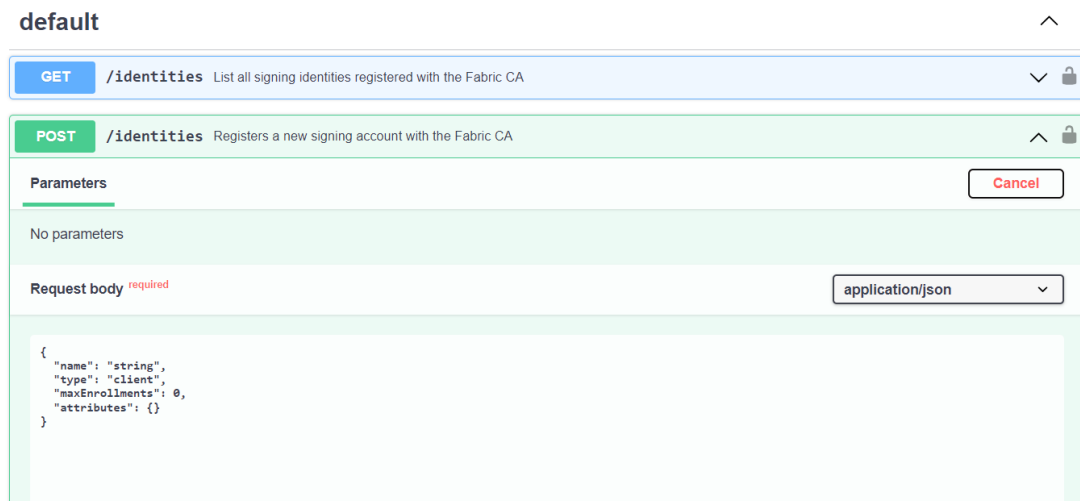


Figure 5.2.38 POST /identities (a)

CHAPTER 5

Paste the code below into the body and click on execute.

```
{
  "name": "user1",
  "type": "client"
}
```

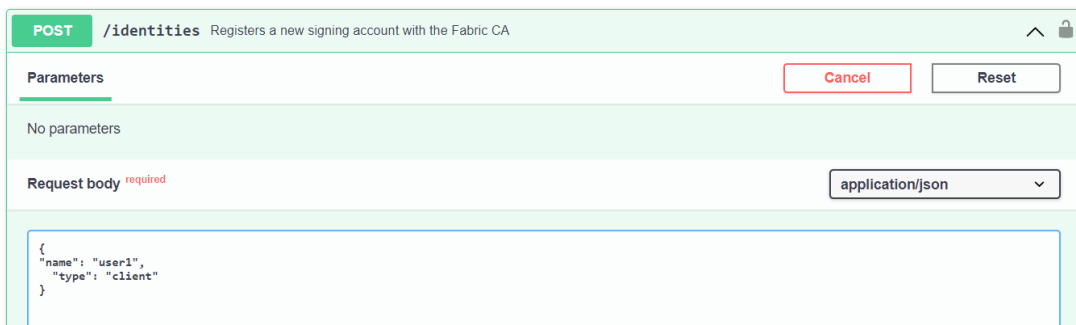


Figure 5.2.39 POST /identities (b)

- After the execution, the Code 200 indicates the successful response from the API.
- The username of the signing account is name, and password is secret which both are generated.



Figure 5.2.40 POST /identities (c) – Server Response

CHAPTER 5

The CA returned a Secret in the API response.

```
{
  "name": "user1",
  "secret": "WxIRaazdwykS"
}
```

Enroll The User

- **POST** to the `/identities/{username}/enroll` route.
- Supply the username in the path and the **secret** in the body.
- Likewise register call, skip the attributes.

```
{
  "secret": " WxIRaazdwykS "
}
```



The screenshot shows a REST client interface for a POST request to the endpoint `/identities/{username}/enroll`. The request body is a JSON object with a `secret` field. The interface includes a 'Parameters' section with a table for path parameters, a 'Request body' section with a dropdown menu for the content type (set to `application/json`), and a 'Cancel' button.

Name	Description
username * required	user1

Request body required: `application/json`

```
{
  "secret": "WxIRaazdwykS"
}
```

Figure 5.2.41 POST `/identities/{username}/enroll` (a)

CHAPTER 5

- It shall return a 200 response from the server. The output shows that the enrolment is successful through success = true line.

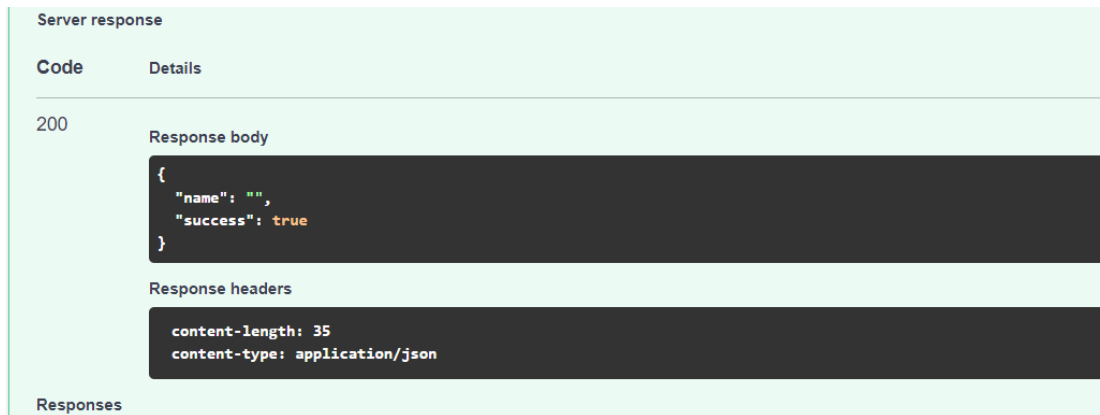


Figure 5.2.42 POST /identities/{username}/enroll (b) – Server Response

Step 10: Query The User Identity Object

- Issue a **GET** on the `/identities/{username}` route.
- Supply the username (user1) in the path and execute it.

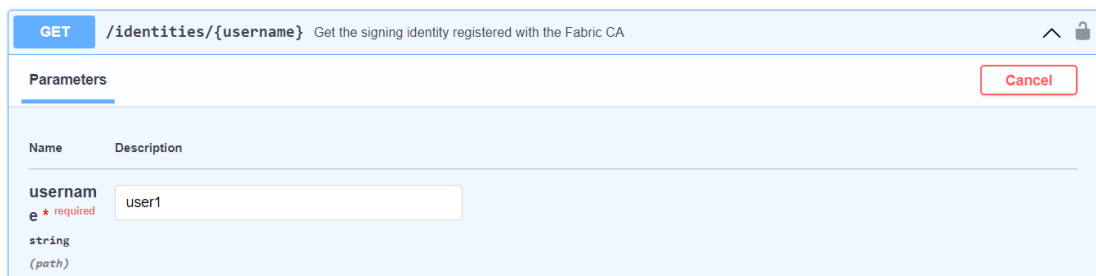


Figure 5.2.43 GET /identities/{username} (a)


```

"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNLakNDQWRDZ0F3SUJBZ0
1VYkRyamk4KzExYkzVHcrWHdmeHhMQkNwNjdBd0NnWUllb1pJemowRUF3SXc
KR3pFWk1CY0dBMVVFQXhNUVptRmljbWxqTFdOaExYTmxjblpsY2pBZUZ3MHIO
REEwTVRnd05qVTVNREJhRncwegpOREEwTVRZd09EUXINREJhTUNFeER6QU5C
Z05WQkFzVEJtTnNhV1Z1ZERFT01Bd0dBMVVFQXhNRmRYTmxjakV3CldUQVRC
Z2NxaGtqT1BRSUJCZ2dxaGtqT1BRTUJCd05DQUFRUm1FY3UxMGRrMEhva3Mxd
3dDNk4zdTBaQnJZaFQKbGFRQINMbG9tNndKN1VVQ0hORW85Vk56bElCTllqODdj
TDFTR3R3K3JGY0t3Qko4R11YdVU5elVvNEhyTUIIbwpNQTRHQTfVZER3RUIvd1F
FQXdJSGdEQU1CZ05WSFJNQkFmOEVBakFBTUIwR0ExVWREZ1FXQkJTtGxPUF
EycHVhCmJ1ak43WitTMVdoL2NhVDY5VEFmQmdOVkhTTUvHREFXZ0JTZ2hxa1
FVQTBLcDROTUVFamxVUUhQSjMrUFhqQXUKQmdOVkhSRUVKekFsZ2IOMU1H
ZHNlZlZtYlxcckxYVXdjVzVrTm1kb05tY3RabUZpY21sakxXNXZaR1V0TURCWQpC
Z2dxQXdrRk1JnY0lBUVJNZXIKaGRIUnljeUk2ZXIKb1ppNUJabVpwYkdsaGRHbHZia
Uk2SWlJc0ltaG1Ma1Z1CmNtOXNiRzFsYm5SSlJDSTZJblZ6WlhJeElpd2lhR111VkhSD
1pTSTZJbU5zYVdWdWRDSjlmVEFLQmdncWhrak8KUFRREFnTk1BREJGQWIFQX
IFWXNJSkEvQUc0NGpqQXAvUU02VIZWcS8xRWhIRUVPaDRqSTRUTkxma3NDS
UIyMQplQ11YUTNuQVFMVxkWe0Y3aGF5dGVKcHpzMW1COVRIQTftL2Y3Z3hpC
i0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K",
  "caCert":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJmVENDQVNPZ0F3SUJBZ0
1VVUo5cUxyVEEx3NGNIZzFDN0FVtYtucVRUSURvd0NnWUllb1pJemowRUF3SXcK
R3pFWk1CY0dBMVVFQXhNUVptRmljbWxqTFdOaExYTmxjblpsY2pBZUZ3MHIOR
EEwTVRnd05qVTVNREJhRncwegpPVEEwTVRvd05qVTVNREJhTUIZeEdUQVhCZ0
5WQkFNVEVHWmhZbkpwWXkxallTMXpaWEoyWlhJd1dUQVRCZ2NxCmhrak9QU
UICQmdncWhrak9QUU1CQndOQ0FBUjkrZTVZQnAva1JRQVVTGh4MFNvMmla
WkNVWEVZd1ZyRlIBZ3gKSTFLa2lJdFZVY0NKM2dpemRoSVJXYTNDczRuY0s5a3
l0bi9EWkUyRVZ3TGtEcnUxbzBVd1F6QU9CZ05WSFE4QgpBZjhFQkFNQ0FRWXdf
Z11EVIiwVEFRSC9CQWd3QmdFQi93SUJBREFkQmdOVkhRNEVGZ1FVb0lhcEVGQ
U5DcWVECIRGQkk1VkVCenlkL2oxNHdDZ11JS29aSXpqMEVBd0lEU0FBd1JRSWh
BT3FhK1BhNmNOMGdEKzFpSWp3L0wrWfCkVXBBRUNrajhQQ2pJVUdDaTUyT1
dBaUFqQXRQUytTVzBtY2xPR2pvcHdKa1ZYSkFqb0c3SFJuVWQ2aldzbS9yLworUT0
9Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K"
}

```


Step 11: Invoke The Deployed Chaincode – Carry Out Transaction (Asset Transfer)

- **POST** to the **/transactions** route.
- Select fly-sync as “false”
- In the body of the call supply values for *signer*, *channel*, *chaincode*, *function* and *args*.
- The argument used for the chaincode parameter is the name of previously set chaincode package, which is “asset_transfer”.
- The method invoked is the CreateAsset method.
- The array for the args are *id*, *color*, *size*, *owner* and *value*.
- Key in the details below, and it the transaction will synchronous:

```
{
  "headers": {
    "type": "SendTransaction",
    "signer": "user1",
    "channel": "mychannel1",
    "chaincode": "asset_transfer"
  },
  "func": "CreateAsset",
  "args": [
    "asset01", "blue", "5", "Nick", "500"
  ],
  "init": false
}
```

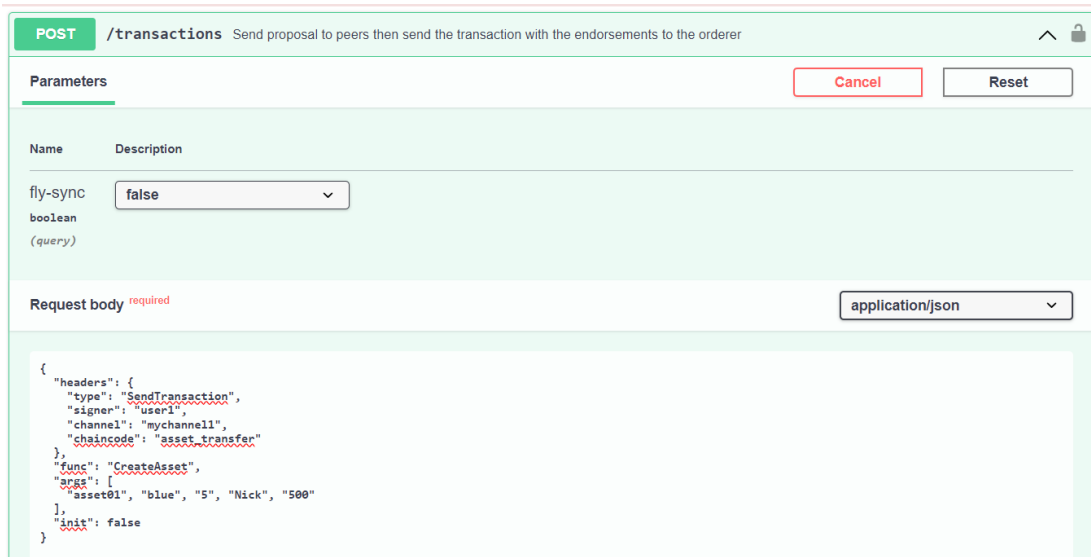


Figure 5.2.45 POST /transactions (a)

- The execution returns 200 and 202 from the server and a `transactionID` is generated stating that the transaction is created.

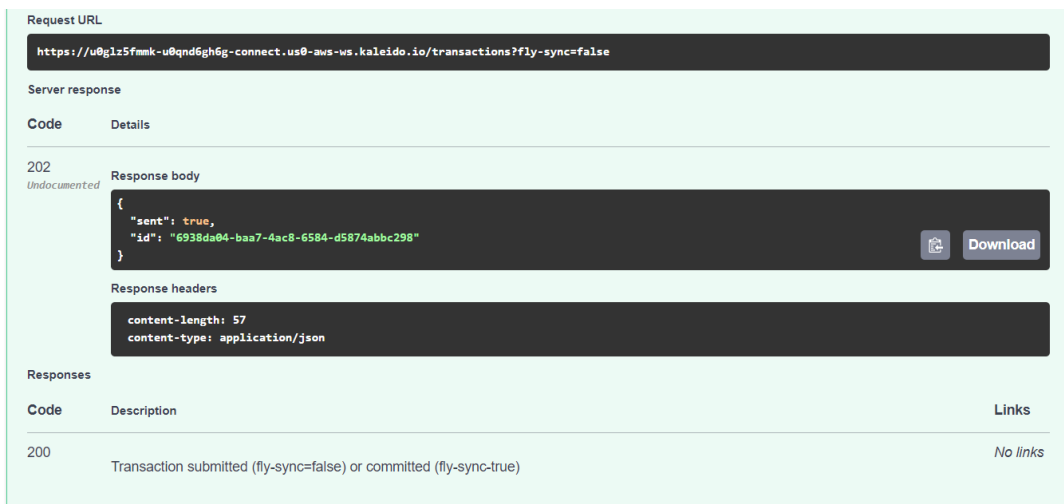


Figure 5.2.46 POST /transactions (b) – Server Response

Invoke The Chaincode in Async Mode

FabConnect offers *async* and *sync* as its two distinct transaction submission types. The *transactionID* will be returned directly in the response for a *sync* submit. We can query against the underlying Apache Kafka topic to obtain the *transactionID*, and an *async* submission will return a *unique receipt ID*.

- **POST** once more to the **/transactions** route with all of the required values and arguments.
- Remember to modify the `id` field for the asset, or else the chaincode will throw an error mentioning that the “asset already exists”.
- Set the `fly-sync` query parameter to `false`.
- A 200 response is received from the server and along with unique message id.

The screenshot shows a REST client interface for a POST request to the `/transactions` endpoint. The description of the endpoint is "Send proposal to peers then send the transaction with the endorsements to the orderer".

Parameters:

Name	Description
fly-sync	false

boolean (query)

Request body required: application/json

```
{
  "headers": {
    "type": "SendTransaction",
    "signer": "user1",
    "channel": "mychannel1",
    "chaincode": "asset_transfer"
  },
  "func": "CreateAsset",
  "args": [
    "asset2", "pink", "5", "John", "1000"
  ],
  "init": false
}
```

Figure 5.2.47 POST /transactions (c) – Async Mode



Figure 5.2.48 POST /transactions (d) – Async Mode Server Response

Query the Async Message ID

To find out the corresponding transaction ID, we need to query the receipt store once we obtain the message ID from an async transaction submission.

- Issue a **GET** on the `/receipts/{receiptId}` route.
- Supply the receipt ID in the path. (`78e87f64-d4c2-4a4d-4be8-3884e37dce34`)
- This should return a similar response body to that you received with a synchronous submission.

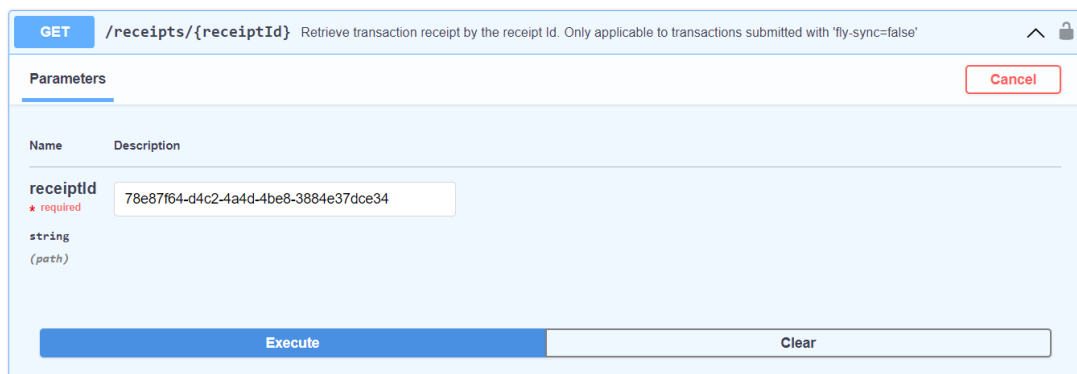


Figure 5.2.49 GET /receipts/{receiptId} (a) - Async Mode

Code Details

200

Response body

```
{
  "_id": "78e87f64-d4c2-4a4d-4be8-3884e37dce34",
  "errorMessage": "Failed to submit: error getting channel response for channel [mychannel1]: Discovery status Code: (11) UNKNOWN. Description: error received from Discovery Server: failed constructing descriptor for chaincodes:<name:\asset_transfer\ > ",
  "headers": {
    "id": "701c7a33-a9a2-4420-4617-a3649bdcca23",
    "requestId": "78e87f64-d4c2-4a4d-4be8-3884e37dce34",
    "requestOffset": "",
    "timeElapsed": 0.001743174,
    "timeReceived": "2024-04-16T09:17:40.902157019Z",
    "type": "Error"
  },
  "receivedAt": 1713431860903,
  "requestPayload": "{\headers\":{\id\":\78e87f64-d4c2-4a4d-4be8-3884e37dce34\",type\":\SendTransaction\",signer\":\user1\",channel\":\mychannel1\",chaincode\":\asset_transfer\",init\":false,func\":\CreateAsset\",args\":[\asset02\",pink\",5\",John\",1000\"]}"
}
```

Response headers

```
content-length: 901
content-type: application/json
```

Responses

Code	Description	Links
200	Receipt returned	No links

Figure 5.2.50 GET /receipts/{receiptId} (b) – Async Mode Server Response

Create an Event Stream

The FabConnect middleware component too supports the dynamic event streaming of events to the backend application. We are required to tell the Kaleido platform about the type of event stream configurations, either webhook or websocket, and the corresponding information, which is the http endpoint or topic, where we want to deliver the data. Webhook will be used.

POST /eventstreams Create a new event stream

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "name": "AssetTransferEvents",
  "type": "webhook",
  "webhook": {
    "url": "https://at123.requestcatcher.com/test",
    "skipVerifyHost": "true"
  }
}
```

Figure 5.2.51 POST /eventstreams (a)

Server response

Code	Details
200	<p>Response body</p> <pre>{ "created": "2024-04-18T09:34:37Z", "id": "es-e5565b06-8cea-4675-6c2f-420250f2d0b3", "name": "AssetTransferEvents", "path": "/eventstreams/es-e5565b06-8cea-4675-6c2f-420250f2d0b3", "suspended": false, "type": "webhook", "batchSize": 1, "batchTimeoutMS": 5000, "errorHandling": "skip", "blockedRetryDelaySec": 30, "webhook": { "url": "https://at123.requestcatcher.com/test", "tlsSkipHostVerify": false, "requestTimeoutSec": 120 }, "timestamps": false, "timestampCacheSize": 1000 }</pre> <p>Response headers</p> <pre>content-length: 520 content-type: application/json</pre>

Responses

Figure 5.2.52 POST /eventstreams (b) – Server Response

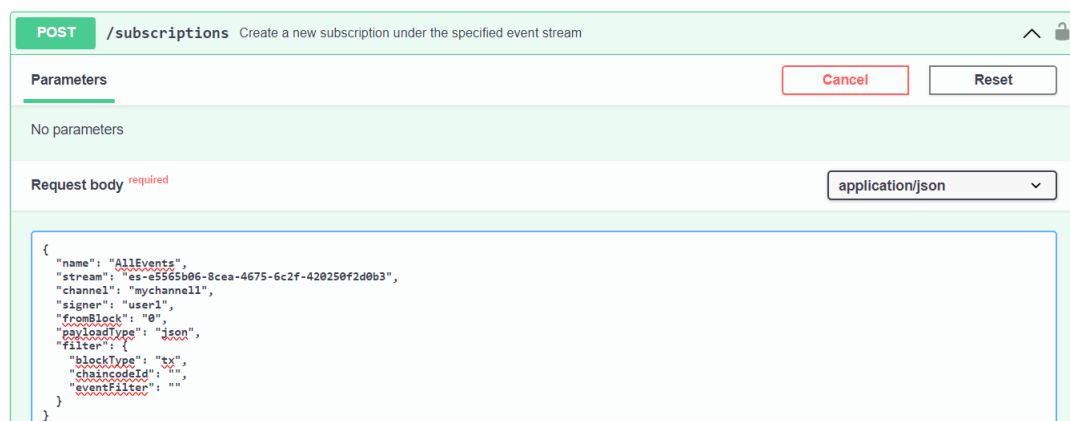
- An event stream ID will be sent in the response body.
- This value is important since it points to the setting that have been made and is necessary for a subscription.

```
{
  "created": "2024-04-18T09:34:37Z",
  "id": "es-e5565b06-8cea-4675-6c2f-420250f2d0b3",
  "name": "AssetTransferEvents",
  "path": "/eventstreams/es-e5565b06-8cea-4675-6c2f-420250f2d0b3",
  "suspended": false,
  "type": "webhook",
  "batchSize": 1,
  "batchTimeoutMS": 5000,
  "errorHandling": "skip",
  "blockedRetryDelaySec": 30,
  "webhook": {
    "url": "https://at123.requestcatcher.com/test",
    "tlsSkipHostVerify": false,
    "requestTimeoutSec": 120
  },
  "timestamps": false,
  "timestampCacheSize": 1000
}
```

Create a Subscription

- **POST** to the **/subscriptions** route.
- In the body of the call supply values for *name*, *stream*, *channel*, *signer*, *fromBlock*, *payloadType* and *blockType*.

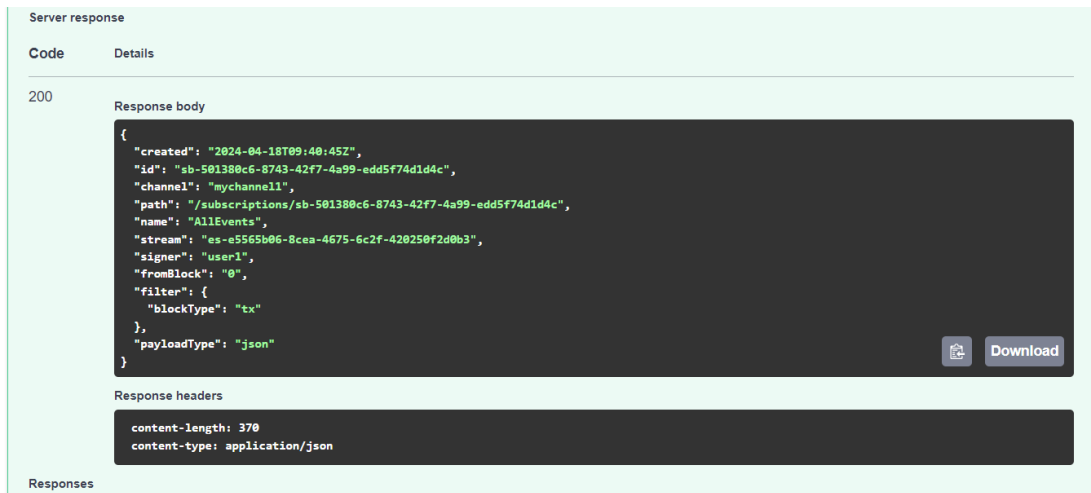
```
{
  "name": "AllEvents",
  "stream": "es-e5565b06-8cea-4675-6c2f-420250f2d0b3",
  "channel": "mychannel1",
  "signer": "user1",
  "fromBlock": "0",
  "payloadType": "json",
  "filter": {
    "blockType": "tx",
    "chaincodeId": "",
    "eventFilter": ""
  }
}
```



The screenshot shows a REST client interface for a POST request to the `/subscriptions` endpoint. The interface includes a title bar with the method `POST` and the endpoint `/subscriptions`, along with a description: "Create a new subscription under the specified event stream". Below the title bar, there are "Parameters" and "Request body" sections. The "Parameters" section is empty, and the "Request body" section is set to `application/json`. The request body is a JSON object with the following structure:

```
{
  "name": "AllEvents",
  "stream": "es-e5565b06-8cea-4675-6c2f-420250f2d0b3",
  "channel": "mychannel1",
  "signer": "user1",
  "fromBlock": "0",
  "payloadType": "json",
  "filter": {
    "blockType": "tx",
    "chaincodeId": "",
    "eventFilter": ""
  }
}
```

Figure 5.2.53 POST /subscriptions (a)



Server response

Code Details

200

Response body

```
{
  "created": "2024-04-18T09:40:45Z",
  "id": "sb-501380c6-8743-42f7-4a99-edd5f74d1d4c",
  "channel": "mychannel1",
  "path": "/subscriptions/sb-501380c6-8743-42f7-4a99-edd5f74d1d4c",
  "name": "AllEvents",
  "stream": "es-e5565b06-8cea-4675-6c2f-420250f2d0b3",
  "signer": "user1",
  "fromBlock": "0",
  "filter": {
    "blockType": "tx"
  },
  "payloadType": "json"
}
```

Response headers

```
content-length: 370
content-type: application/json
```

Responses

Figure 5.2.54 POST /subscriptions (b) – Server Response

5.3 Implementation Using Ubuntu Linux Virtual Machine

5.3.1 Generating Network and Invoking Smart Contracts

Import Repository from GitHub

In this case, the asset transfer smart contract is executed on GitHub using a repository from the Hyperledger fabric. Use the following command to clone the fabric-samples repository:

```
git clone https://github.com/hyperledger/fabric-samples.git
```

Navigate to the directory using the command below.

```
cd ~/fabric-samples/test-network
```

Creating a Channel Using Hyperledger Fabric

In the directory, deploy a new instance using the script prepared. Can check the availability of the script using `ls -l` method.

```
./network.sh up createChannel -c mychannel
```

```
admin-ooiyo@admin-ooiyo:~/fabric-samples/test-network$ ./network.sh createChannel
Using docker and docker-compose
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database
'leveldb with crypto from 'cryptogen'
Bringing up network
LOCAL VERSION=v2.5.9
DOCKER IMAGE VERSION=v2.5.9
/home/admin-ooiyo/fabric-samples/test-network/./bin/cryptogen
generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
[*] Running 7/7
  * Network fabric_test          Created          0.2s
  * Volume "compose_orderer.example.com" Created          0.0s
  * Volume "compose_peer0.org1.example.com" Created          0.0s
  * Volume "compose_peer0.org2.example.com" Created          0.0s
  * Container peer0.org2.example.com Started          0.3s
  * Container orderer.example.com Started          0.3s
  * Container peer0.org1.example.com Started          0.3s
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS
NAMES
8620f9ed10c1   hyperledger/fabric-peer:latest      "peer node start"       2 seconds ago Up Less than a second
0.0.0.0:9051->9051/tcp, :::9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp
peer0.org2.example.com
7f78ca9d87e4   hyperledger/fabric-peer:latest      "peer node start"       2 seconds ago Up Less than a second
0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp
peer0.org1.example.com
4060907a5457   hyperledger/fabric-orderer:latest   "orderer"               2 seconds ago Up Less than a second
0.0.0.0:7050->7050/tcp, :::7050->7050/tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp
orderer.example.com
95a180a90170   hello-world                          "/hello"                 28 minutes ago Exited (0) 28 minutes ago
nice_lichterman
```

Figure 5.3.1.1 Creating Channel using network.sh script

CHAPTER 5

Figure below shows that the channel has been successfully created and prompt the dialog:

Channel 'mychannel' joined

```
2024-09-06 09:58:04.269 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2024-09-06 09:58:04.290 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
admin-ooiyo@admin-ooiyo:~/fabric-samples/test-network$
```

Figure 5.3.1.2 Channel Successfully Joined

Initiating a Chaincode on The Channel

```
./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl
```

```
admin-ooiyo@admin-ooiyo:~/fabric-samples/test-network$ ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basi
c/chaincode-go -ccl go
Using docker and docker-compose
Deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: basic
- CC_SRC_PATH: ../asset-transfer-basic/chaincode-go
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
- CC_SEQUENCE: auto
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FC: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
executing with the following
- CC_NAME: basic
- CC_SRC_PATH: ../asset-transfer-basic/chaincode-go
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
Vendoring Go dependencies at ../asset-transfer-basic/chaincode-go
~/fabric-samples/asset-transfer-basic/chaincode-go ~/fabric-samples/test-network
~/fabric-samples/test-network
Finished vendoring Go dependencies
+ '[' false = true ']'
+ peer lifecycle chaincode package basic.tar.gz --path ../asset-transfer-basic/chaincode-go --lang golang --label b
asic_1.0
+ res=0
Chaincode is packaged
```

Figure 5.3.1.3 Deploying Chaincode to Instances (a)

```

admin-ooiyo@admin-ooiyo: ~/fabric-samples/test-network
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Checking the commit readiness of the chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name basic --version 1.0 --sequence 1 --out
put json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": true
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org2 on channel 'mychannel'
Using organization 1
Using organization 2
+ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile
/home/admin-ooiyo/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.c
om-cert.pem --channelID mychannel --name basic --peerAddresses localhost:7051 --tlsRootCertFiles /home/admin-ooiyo/
fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/tlsca/tlsca.org1.example.com-cert.pem
--peerAddresses localhost:9051 --tlsRootCertFiles /home/admin-ooiyo/fabric-samples/test-network/organizations/peerO
rganizations/org2.example.com/tlsca/tlsca.org2.example.com-cert.pem --version 1.0 --sequence 1
+ res=0
2024-09-06 10:21:00.161 UTC 0001 INFO [chaincodeCmd] ClientWait -> txid [52c5ca1018b4fb2b33fa9a6176f541e0373b02cd7c
6fd4b5ff1f097db70d5327] committed with status (VALID) at localhost:9051
2024-09-06 10:21:00.168 UTC 0002 INFO [chaincodeCmd] ClientWait -> txid [52c5ca1018b4fb2b33fa9a6176f541e0373b02cd7c
6fd4b5ff1f097db70d5327] committed with status (VALID) at localhost:7051
Chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escv, Validation Plugin: vscv, Approvals: [Org1MSP: true, Org2MSP: t
rue]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escv, Validation Plugin: vscv, Approvals: [Org1MSP: true, Org2MSP: t
rue]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
admin-ooiyo@admin-ooiyo:~/fabric-samples/test-network$

```

Figure 5.3.1.4 Deploying Chaincode to Instances (b)

Interacting with The Network

Operate it from test-network directory. Find the `peer` binaries in the `bin` folder of the `fabric-samples` repository. Use the following command to add those binaries to the CLI Path:

```
export PATH=${PWD}/../bin:$PATH
```

`FABRIC_CFG_PATH` are needed to point to the `core.yaml` file in the `fabric-samples` repository:

```
export FABRIC_CFG_PATH=$PWD/../../config/
```

Set the environment variables to operate the `peer` CLI as Org1:

Bachelor of Information Systems (Honours) Digital Economy Technology
Faculty of Information and Communication Technology (Kampar Campus), UTAR

```
# Environment variables for Org1

export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export
CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.
example.com/peers/peer0.org1.example.com/tls/ca.crt
export
CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.exa
mple.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051
```

Run the following command to initialize the ledger with assets:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
"${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.exa
mple.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --
peerAddresses localhost:7051 --tlsRootCertFiles
"${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.e
xample.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles
"${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.e
xample.com/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'
```

Output shall appear as below, indicating success invocation:

-> INFO 001 Chaincode invoke successful. result: status:200

```
admin-ooiyo@admin-ooiyo:~/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostname
Override orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer
.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsR
ootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --pe
erAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.
org2.example.com/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'
2024-09-06 10:25:07.592 UTC 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result:
status:200
```

Figure 5.1.3.5 Initializing Ledger with Assets

Run the following command to get the list of assets that were added to the channel ledger:

```
peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}'
```

```
admin-ooi@admin-ooi:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}'
[{"AppraisedValue":300,"Color":"blue","ID":"asset1","Owner":"Tomoko","Size":5}, {"AppraisedValue":400,"Color":"red","ID":"asset2","Owner":"Brad","Size":5}, {"AppraisedValue":500,"Color":"green","ID":"asset3","Owner":"Jin Soo","Size":10}, {"AppraisedValue":600,"Color":"yellow","ID":"asset4","Owner":"Max","Size":10}, {"AppraisedValue":700,"Color":"black","ID":"asset5","Owner":"Adriana","Size":15}, {"AppraisedValue":800,"Color":"white","ID":"asset6","Owner":"Michel","Size":15}]
```

Figure 5.1.3.6 Query the Ledger on Peer0.Org1

Output:

```
[
  {"AppraisedValue":300,"Color":"blue","ID":"asset1","Owner":"Tomoko","Size":5},
  {"AppraisedValue":400,"Color":"red","ID":"asset2","Owner":"Brad","Size":5},
  {"AppraisedValue":500,"Color":"green","ID":"asset3","Owner":"Jin Soo","Size":10},
  {"AppraisedValue":600,"Color":"yellow","ID":"asset4","Owner":"Max","Size":10},
  {"AppraisedValue":700,"Color":"black","ID":"asset5","Owner":"Adriana","Size":15},
  {"AppraisedValue":800,"Color":"white","ID":"asset6","Owner":"Michel","Size":15}
]
```

Invoking a Chaincode

When a member of the network wishes to move or modify an asset on the ledger, chaincodes are triggered. To modify the owner of an asset on the ledger, use the asset-transfer (basic) chaincode as follows:

```
peer chaincode invoke -o localhost:7050 --ordererTLShostnameOverride
orderer.example.com --tls --cafile
"${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses
localhost:7051 --tlsRootCertFiles
"${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles
"${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"TransferAsset","Args":["asset6","Christopher"]}'
```

```
admin-ooi@admin-ooi:~/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"TransferAsset","Args":["asset6","Christopher"]}'
2024-09-06 10:28:22.581 UTC 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200 payload:"Michel"
```

Figure 5.1.3.7 Invoking Chaincode

If the command is successful, the following response shall appear:

```
2024-09-06 10:28:22.581 UTC 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery ->
Chaincode invoke successful. result: status:200 payload:"Michel"
```

We may use a different query to check how the chaincode invocation affected the assets on the blockchain ledger after it has been executed. We can now query the chaincode executing on the Org2 peer as we have already questioned the Org1 peer. To function as Org2, set the following environment variables:

```
# Environment variables for Org2

export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org2MSP"
export
CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
export
CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
export CORE_PEER_ADDRESS=localhost:9051
```

We can now query the asset-transfer (basic) chaincode running on peer0.org2.example.com:

```
peer chaincode query -C mychannel -n basic -c '{"Args":["ReadAsset","asset6"]}'
```

```
admin-ooi@admin-ooi:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n basic -c '{"Args":["ReadAsset","asset6"]}'
{"AppraisedValue":800,"Color":"white","ID":"asset6","Owner":"Christopher","Size":15}
```

Figure 5.1.3.8 Query the Ledger on Peer0.Org2

The result will show that "asset6" was transferred to Christopher:

```
{"AppraisedValue":800,"Color":"white","ID":"asset6","Owner":"Christopher","Size":15}
```

Bring Down the Network

```
./network.sh down
```

The command will remove the chaincode images from your Docker Registry, stop and remove the node and chaincode containers, and delete the organization's crypto content.

Bring up The Network with Certificate Authorities

```
./network.sh up -ca
```

```
admin-ooiyo@admin-ooiyo:~/fabric-samples/test-network$ ./network.sh up -ca
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto
from 'Certificate Authorities'
LOCAL_VERSION=v2.5.9
DOCKER_IMAGE_VERSION=v2.5.9
CA_LOCAL_VERSION=v1.5.12
CA_DOCKER_IMAGE_VERSION=v1.5.12
Generating certificates using Fabric CA
[+] Running 4/4
✔ Network fabric_test Created 0.18s
✔ Container ca_oig2 Started 0.18s
✔ Container ca_orderer Started 0.18s
```

Figure 5.1.3.9 Bring up CAs for Organizations in the Network

The test network registers nodes and user identities with each organization's CA by using the Fabric CA client. The script then creates an MSP folder for each identity using the enroll command. Each identity's certificate and private key, as well as information on the identity's membership and function within the company running the CA, are contained in the MSP folder.

Examine MSP folder of the Org1 admin user:

```
tree organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/
```

```

admin-ooi@admin-ooi:~/fabric-samples/test-network$ tree organizations/peerOrganizations/org1.example.com/users/
Admin@org1.example.com/
locales-launch: Data of en_US locale not found, generating, please wait...
organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/
├── msp
│   ├── cacerts
│   │   └── localhost-7054-ca-org1.pem
│   ├── config.yaml
│   ├── IssuerPublicKey
│   ├── IssuerRevocationPublicKey
│   ├── keystore
│   │   └── 0053c0f1db4717a527f9058b072fb0a89474c1b11ba8bf45cbc44bb24e3fcdb4_sk
│   ├── signcerts
│   │   └── cert.pem
│   └── user
5 directories, 6 files

```

Figure 5.1.3.10 MSP Folder of Org1 Admin User

Output:

```

organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/
├── msp
│   ├── cacerts
│   │   └── localhost-7054-ca-org1.pem
│   ├── config.yaml
│   ├── IssuerPublicKey
│   ├── IssuerRevocationPublicKey
│   ├── keystore
│   │   └── 0053c0f1db4717a527f9058b072fb0a89474c1b11ba8bf45cbc44bb24e3fcdb4_sk
│   ├── signcerts
│   │   └── cert.pem
│   └── user

```


5.3.2 Initializing Hyperledger Firefly CLI

Creating a Firefly Development Stack

To create an Ethereum based stack, run:

```
ff init ethereum
```

To create a Fabric based stack, run:

```
ff init fabric
```

Enter a unique stack name:

```
stack name: <stack_name>
```

```
stack name: my_supply_chain_managment
```

Enter the preferred number of members for your stack:

```
number of members: 4
```

```
kityann@Starfishhh:/mnt/c/Users/kitya$ ff init fabric
initializing new FireFly stack...
stack name: my_supply_chain_management
You selected my_supply_chain_management
number of members: 4
Stack 'my_supply_chain_management' created!
To start your new stack run:

ff start my_supply_chain_management

Your docker compose file for this stack can be found at: /home/kityann
/.firefly/stacks/my_supply_chain_management/docker-compose.yml
```

Figure 5.3.2.1 Creating Firefly Development Stack

Starting the Stack

To start your stack simply run:

```
ff start <stack_name>
```

```
ff start my_supply_chain_management
```

```
kityann@Starfishhh:/mnt/c/Users/kitya$ ff start my_supply_chain_management
this will take a few seconds longer since this is the first time you're runni
ng this stack...
done

Web UI for member '0': http://127.0.0.1:5000/ui
Swagger API UI for member '0': http://127.0.0.1:5000/api
Sandbox UI for member '0': http://127.0.0.1:5108

Web UI for member '1': http://127.0.0.1:5001/ui
Swagger API UI for member '1': http://127.0.0.1:5001/api
Sandbox UI for member '1': http://127.0.0.1:5208

Web UI for member '2': http://127.0.0.1:5002/ui
Swagger API UI for member '2': http://127.0.0.1:5002/api
Sandbox UI for member '2': http://127.0.0.1:5308

Web UI for member '3': http://127.0.0.1:5003/ui
Swagger API UI for member '3': http://127.0.0.1:5003/api
Sandbox UI for member '3': http://127.0.0.1:5408

To see logs for your stack run:

ff logs my_supply_chain_management

kityann@Starfishhh:/mnt/c/Users/kitya$ |
```

Figure 5.3.2.2 Starting Firefly Stack

Web UI Interface – Firefly Explorer

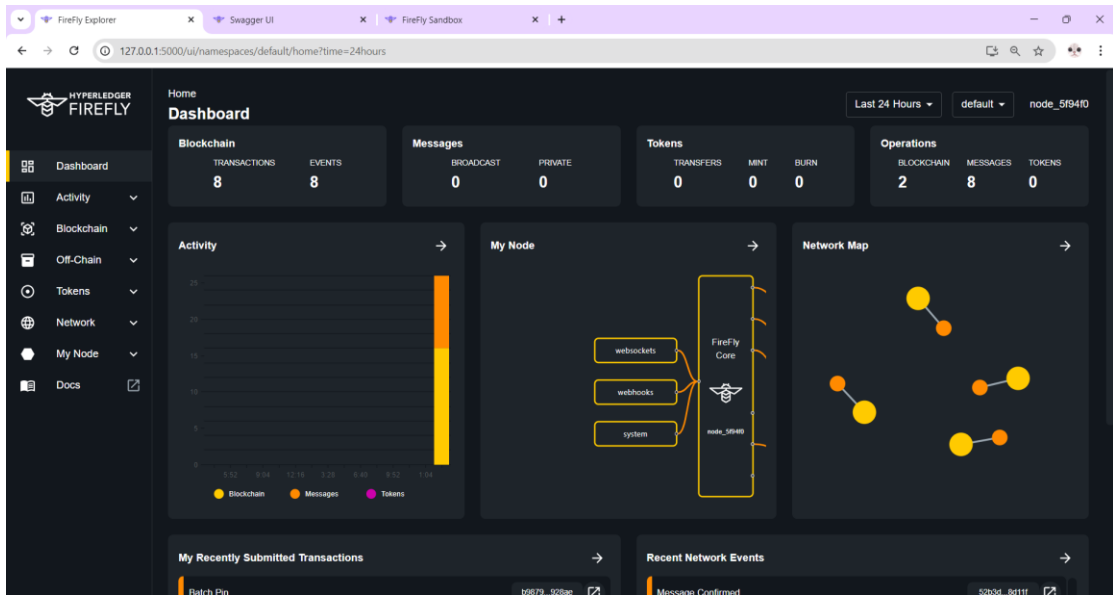


Figure 5.3.2.3 Web UI Interface – Firefly Explorer (a)

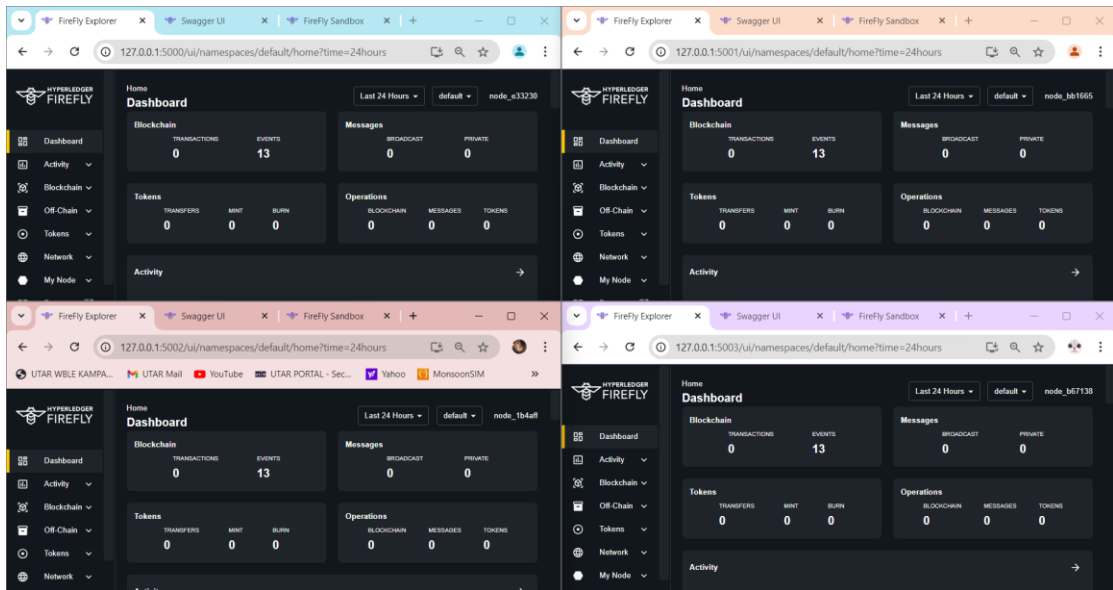


Figure 5.3.2.4 Web UI Interface – Firefly Explorer (b)

Firefly Sandbox

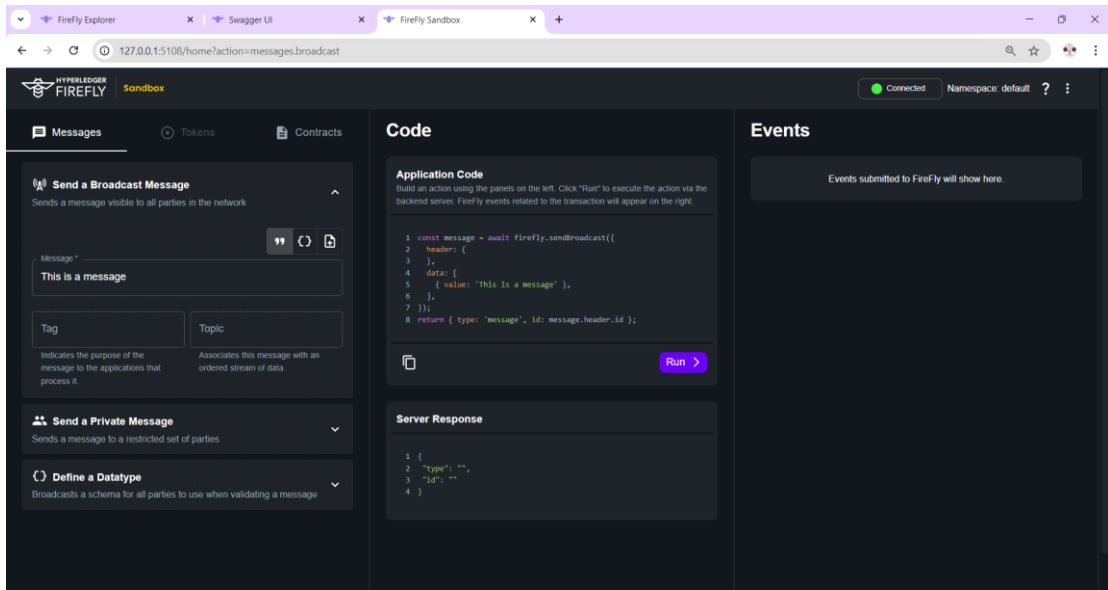


Figure 5.3.2.5 Firefly Sandbox (a)

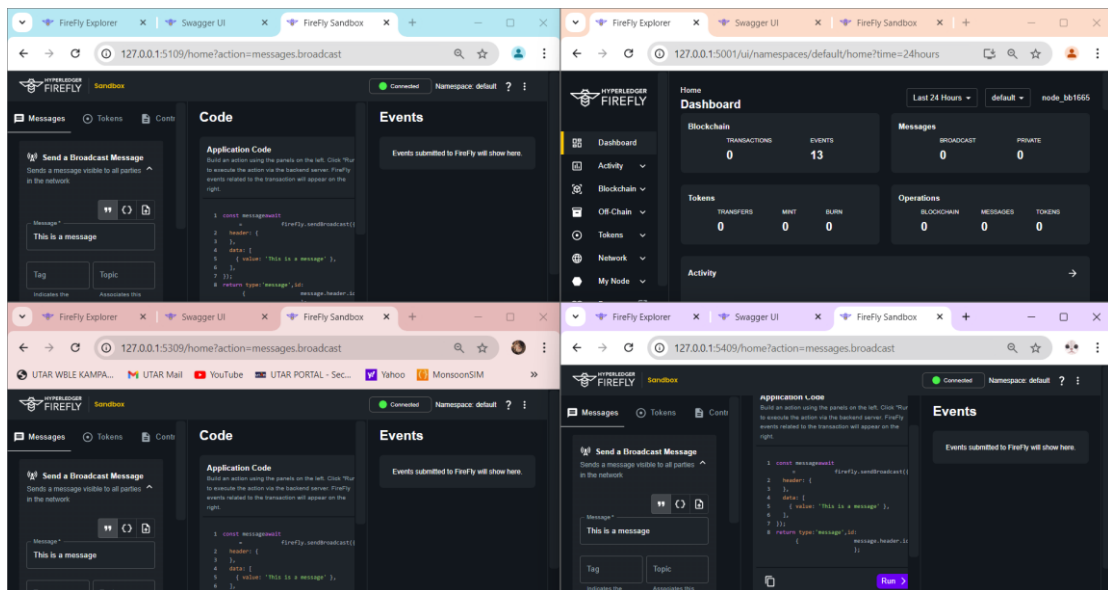


Figure 5.3.2.6 Firefly Sandbox (b)

Swagger API UI

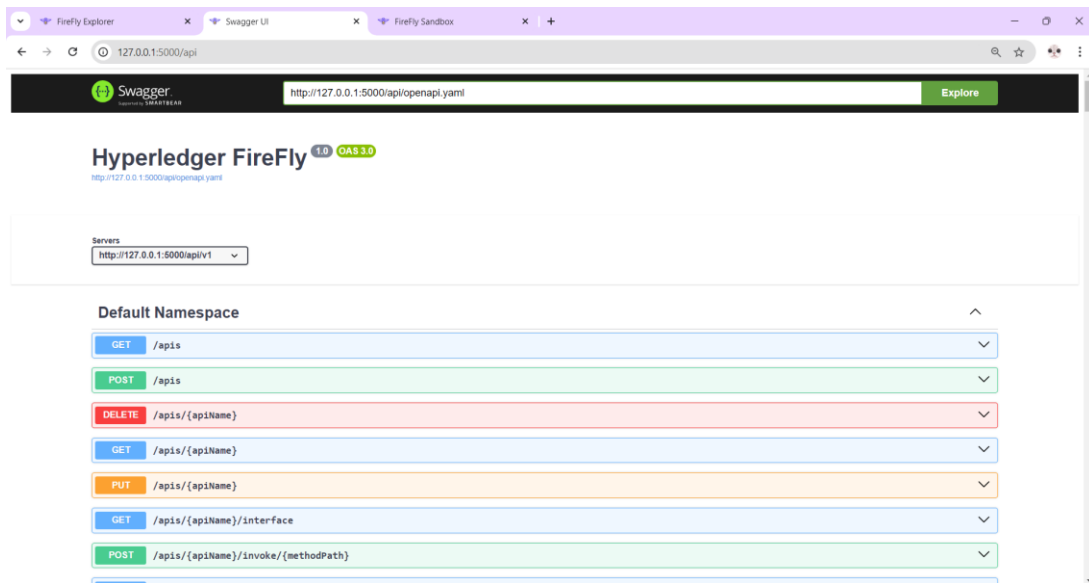


Figure 5.3.2.7 Swagger API UI (a)

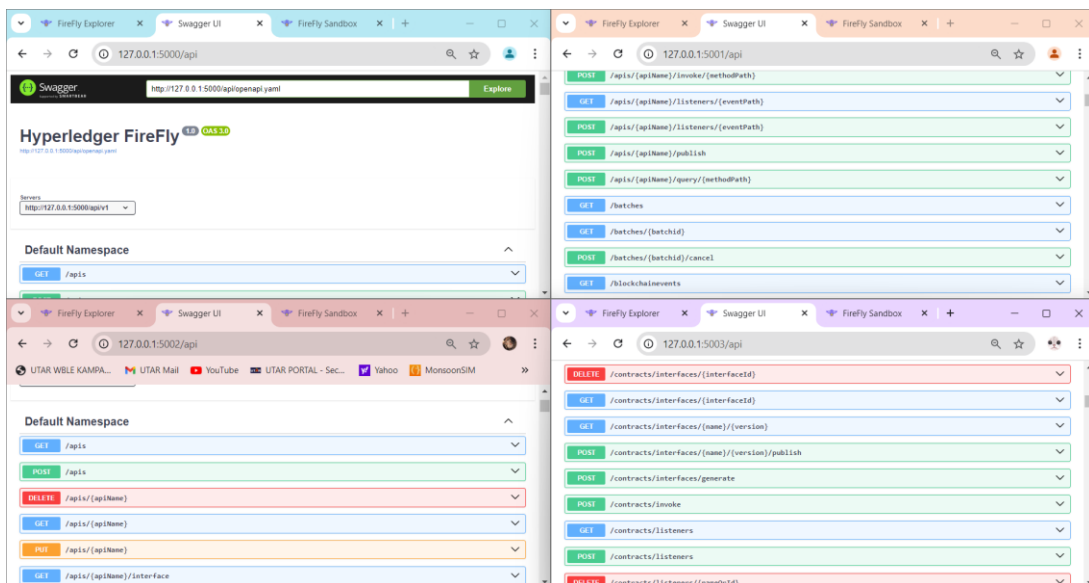


Figure 5.3.2.8 Swagger API UI (b)

Broadcasting Message using the Sandbox Environment

Scenario 1:

Member 0 in the channel wants to send a broadcast message to all the members within the channel. The member wants to acknowledge everyone about the closure of order acceptance due to holidays.

Enter details below on the message, tag and topic column:

****Order Inquiries****

Dear all,

Please take note that we will be closing order on 8 September 2024 due to holidays.

Tag: `order`

Topic: `Holiday`

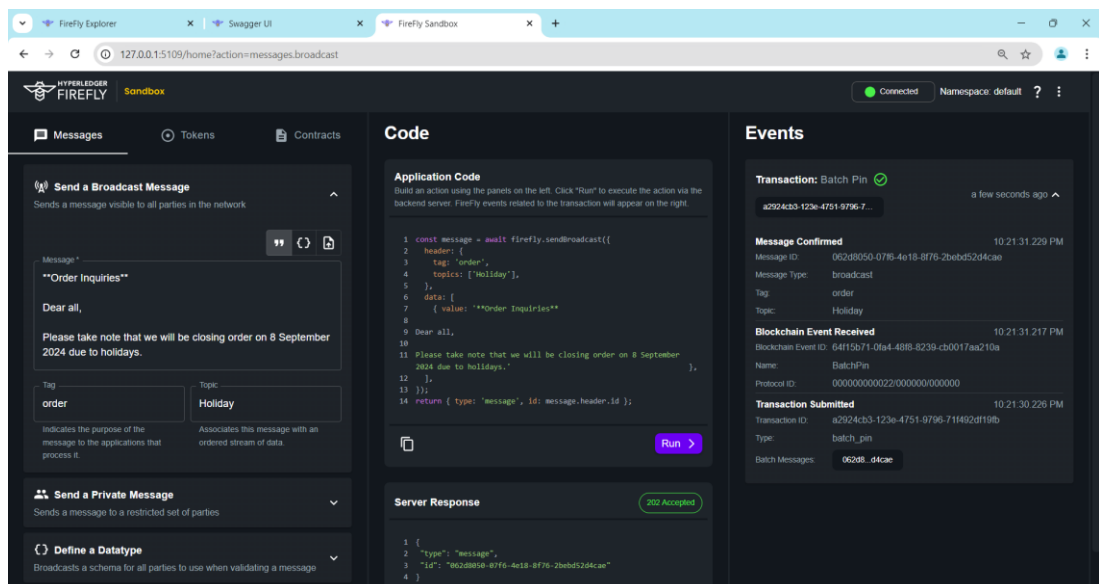


Figure 5.3.2.9 Member 0 Sending Broadcast Message

CHAPTER 5

After entering all the details and messages, click the blue “run” button. A response of 202 has returned indicating a successive of sending messages to all members within the channel.

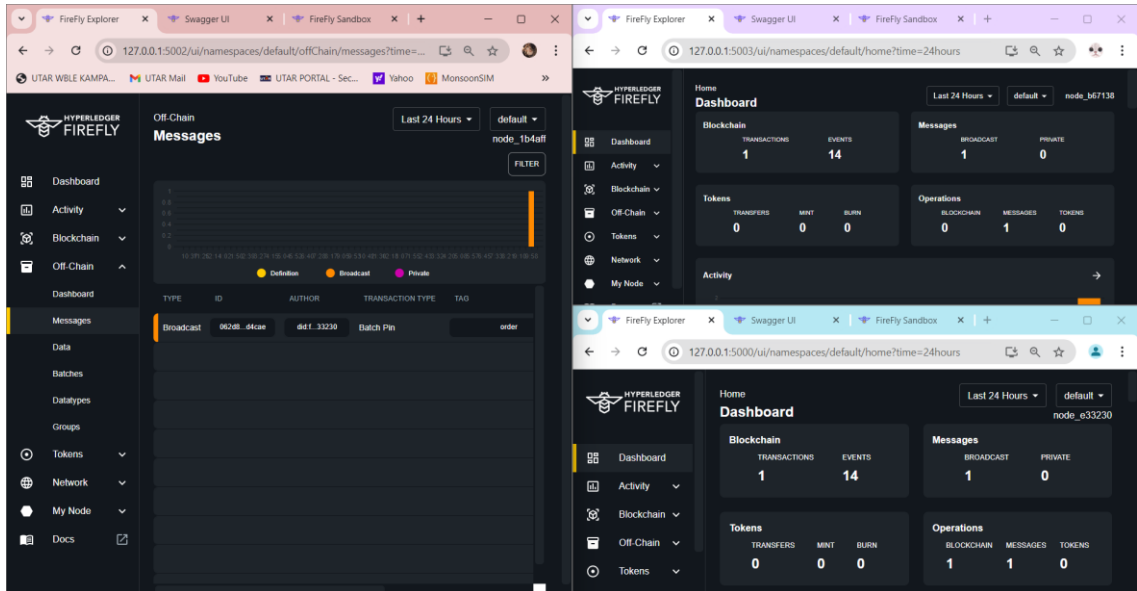


Figure 5.3.2.10 Broadcast Message Received

All members (Member 1, 2, and 3) will received the message from Member 0, by referring the Firefly Explorer, on the Off-Chain > Messages.

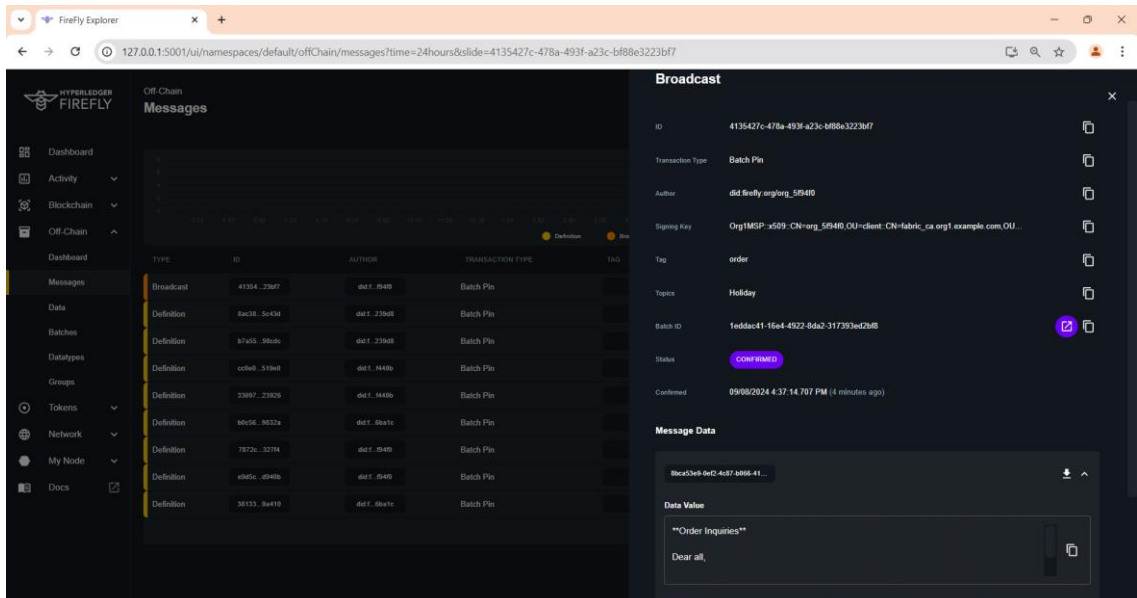


Figure 5.3.2.11 Member 1 Firefly Explorer – Broadcast Message

Sending Private Messages using the Sandbox Environment

Bachelor of Information Systems (Honours) Digital Economy Technology
Faculty of Information and Communication Technology (Kampar Campus), UTAR

CHAPTER 5

Scenario 2:

Member 1 wants to send a private message to member 0 within the channel. The member wants to provide a supplier performance feedback to member 0 to take corrective actions on it. Other members are not acknowledged and does not have permission of viewing this message.

Enter details below on the message, tag and topic column:

Supplier Performance Feedback:

We have noted that recent shipments from your facility have had a defect rate of 3%. Please review and take corrective action.

Tag: [feedback](#)

Topic: [performance](#)

The screenshot displays the Firefly Explorer interface. On the left, the 'Messages' panel shows the 'Send a Private Message' form with the following details: Message: 'Supplier Performance Feedback. We have noted that recent shipments from your facility have had a defect rate of 3%. Please review and take corrective action.'; Recipients: 'did:firefly.org/org_59410'; Tag: 'feedback'; Topic: 'performance'. The 'Code' panel in the center shows the application code for sending a private message. The 'Events' panel on the right shows a 'Transaction Submitted' event with a 'Batch Pin' and a 'Message Confirmed' event with details for the message ID, type, tag, and topic.

Figure 5.3.2.12 Member 1 Sending Private Message

CHAPTER 5

After entering all the details and messages, click the blue “run” button. A response of 202 has returned indicating a successive of sending messages only to member 0 within the channel. Other members are not authorized to view the message.

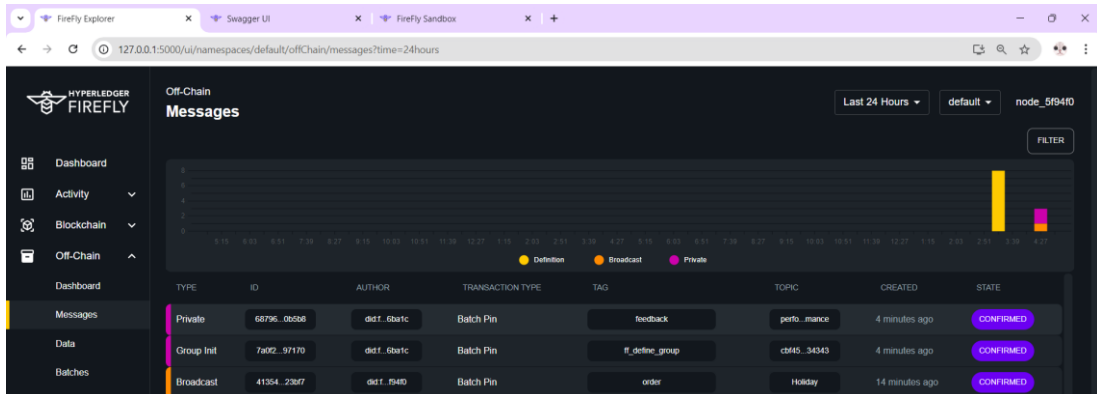


Figure 5.3.2.13 Member 0 Firefly Explorer – Private Message

By looking at the Firefly Explorer of Member 0, we can clearly see that the private message is received. Whereas Member 2 and 3 does not receive the private message send by Member 1. This has ensured the security measures of the entire message process by using the Hyperledger Fabric.

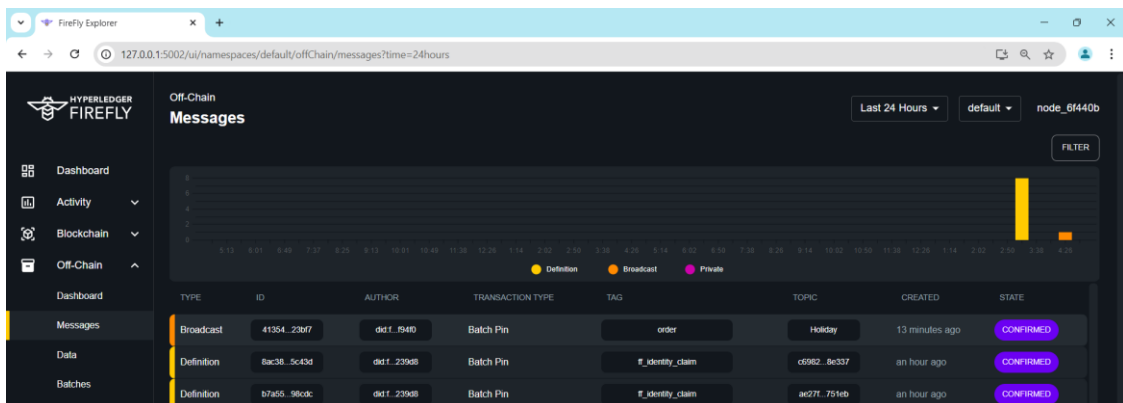


Figure 5.3.2.14 Member 2 Firefly Explorer – Private Message

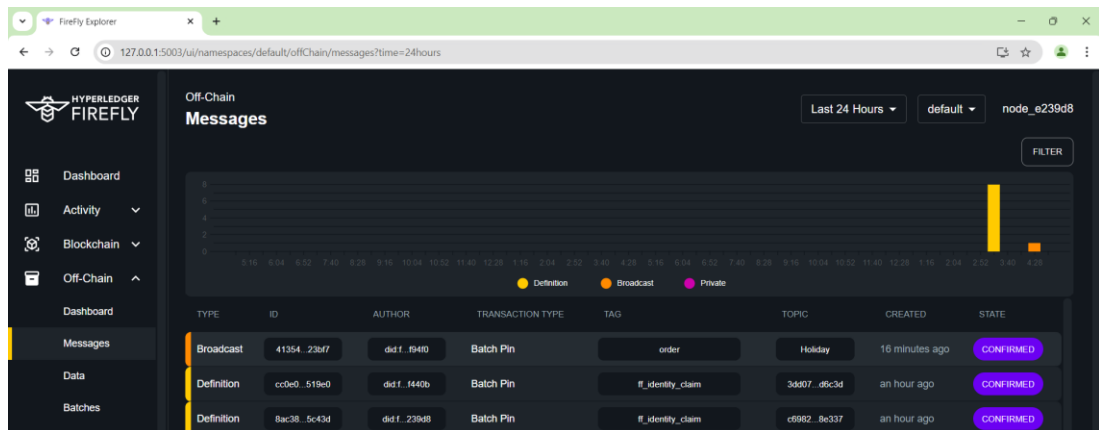


Figure 5.3.2.15 Member 3 Firefly Explorer – Private Message

Downloading files from other members in Firefly Explorer

Scenario 3:

Member 0 sends a private file to member 1, to update member 1 on the latest review on the performance feedback.

Enter details below on the message, tag and topic column:

Tag: [feedback](#)

Topic: [performance](#)

After uploading the file, click the blue “run” button. A response of 202 has returned indicating a successive of sending messages only to member 1 within the channel. Other members are not authorized to view the message.

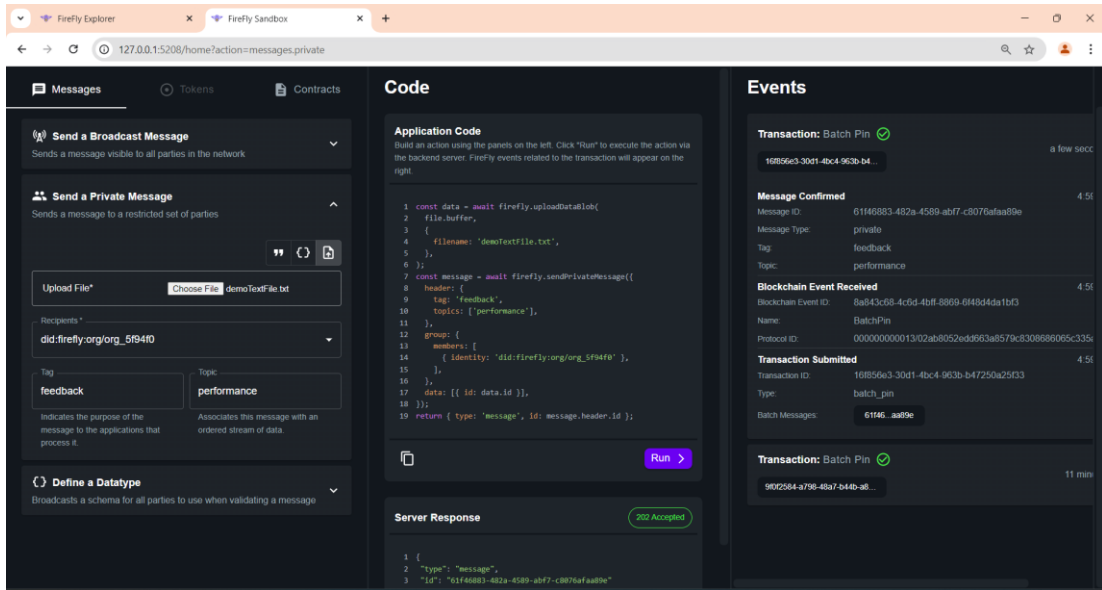


Figure 5.3.2.16 Member 0 Sending Private File

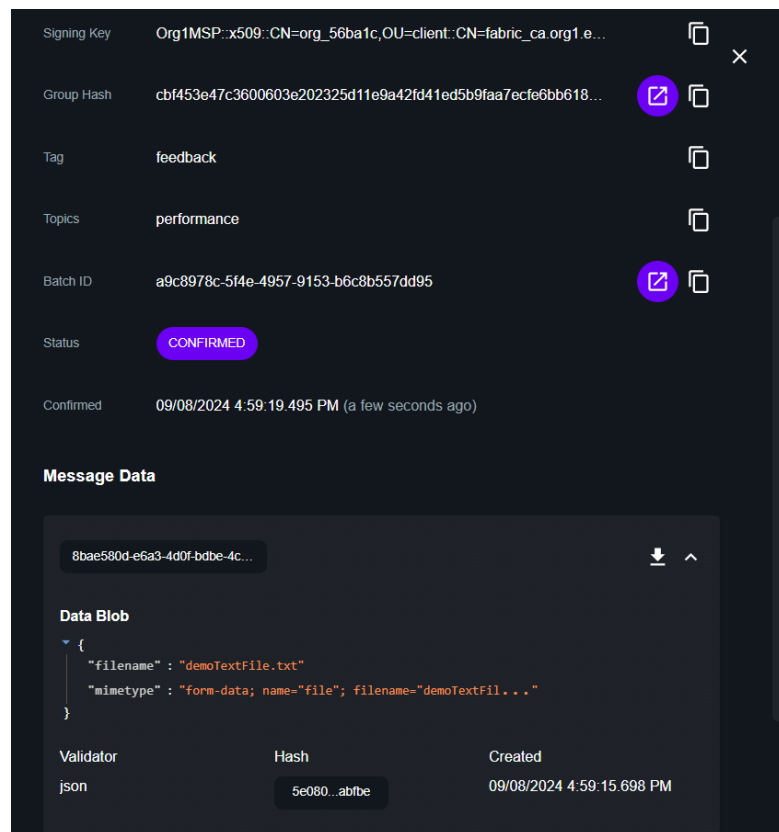


Figure 5.3.2.17 Member 1 – Download the .txt File

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

The system testing was conducted using a combination of the Kaleido platform and a Virtual Machine (VM) running Ubuntu 20.04.6 (Server-amd64-VB7.0.8) as a sandbox environment. Various performance metrics were evaluated, focusing on system functionality, compatibility, and resource efficiency. Windows are used in testing the environment. In this case, WSL are required and installed by opening the desktop terminal.

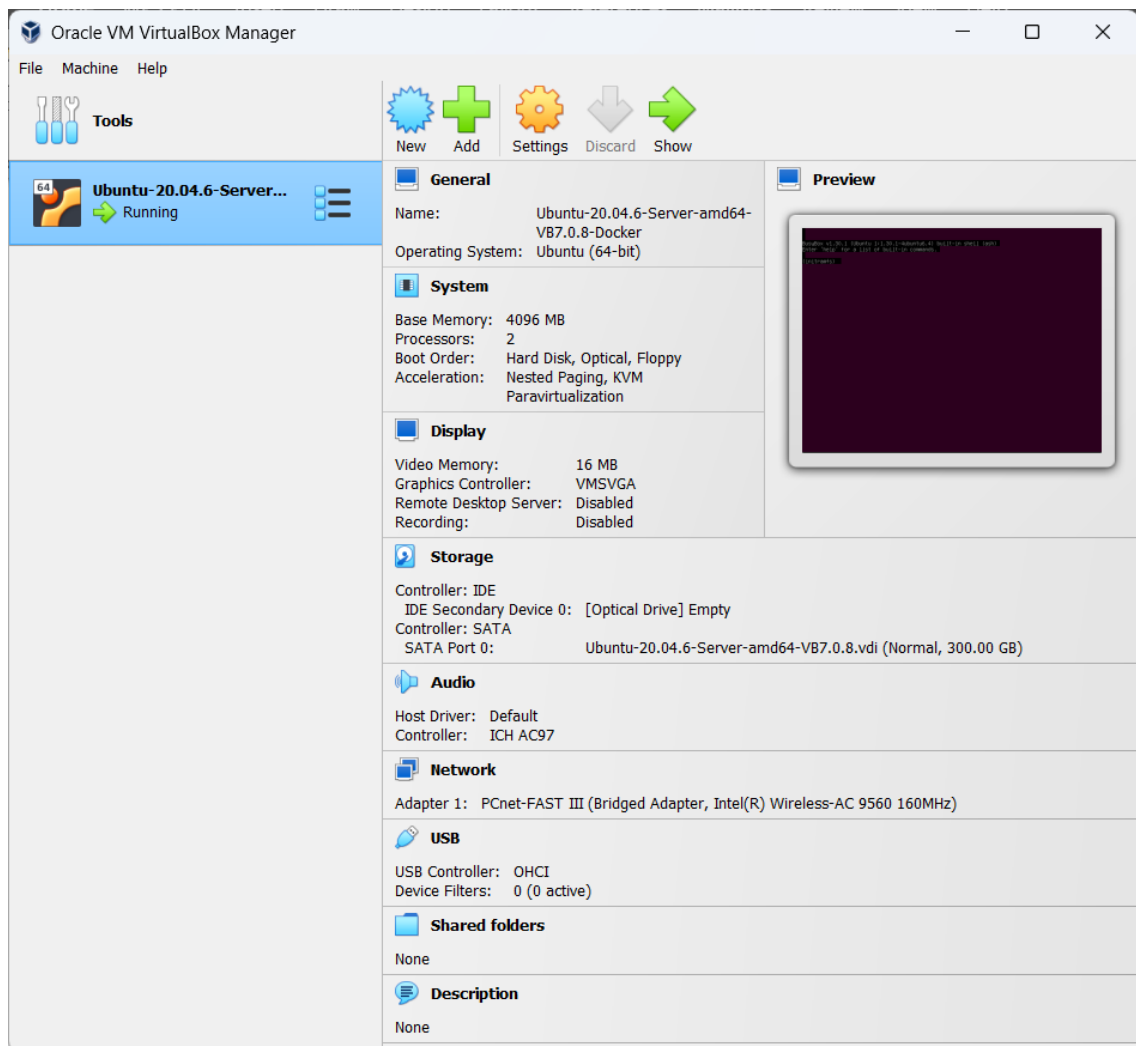


Figure 6.1.1 VM - Ubuntu Version 20.04.6

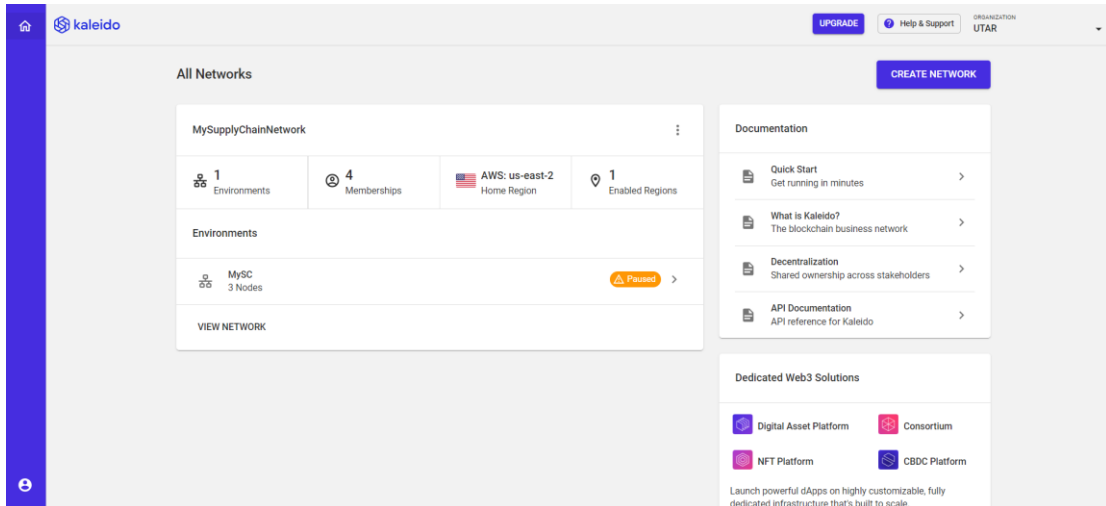


Figure 6.1.2 Kaleido Platform

```

kityann@Starfishhh: /mnt/c/L
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kitya>wsl
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/home/kityann/.hushlogin file.
kityann@Starfishhh: /mnt/c/Users/kitya$ |
  
```

Figure 6.1.3 WSL

Pre-requisite technologies, including Docker, Docker Compose, Golang, JavaScript, and Node.js, were installed to support the blockchain infrastructure, specifically Hyperledger Fabric, which was utilized to ensure the security aspect of the supply chain management system.

```

kityann@Starfishhh: /mnt/c/L x + v
kityann@Starfishhh:/mnt/c/Users/kitya$ docker version
Client: Docker Engine - Community
Version:      27.2.0
API version:  1.47
Go version:   go1.21.13
Git commit:   3ab4256
Built:        Tue Aug 27 14:15:13 2024
OS/Arch:      linux/amd64
Context:      default

Server: Docker Engine - Community
Engine:
Version:      27.2.0
API version:  1.47 (minimum version 1.24)
Go version:   go1.21.13
Git commit:   3ab5c7d
Built:        Tue Aug 27 14:15:13 2024
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      1.7.21
GitCommit:    472731909fa34bd7bc9c087e4c27943f9835f111
runc:
Version:      1.1.13
GitCommit:    v1.1.13-0-g58aa920
docker-init:
Version:      0.19.0
GitCommit:    de40ad0
kityann@Starfishhh:/mnt/c/Users/kitya$ |

```

Figure 6.1.4 Docker Version

```

kityann@Starfishhh:/mnt/c/Users/kitya$ docker compose version
Docker Compose version v2.29.2

```

Figure 6.1.5 Docker Compose Version

```

kityann@Starfishhh:/mnt/c/Users/kitya$ go version
go version go1.18.1 linux/amd64

```

Figure 6.1.6 Go Version

```

kityann@Starfishhh:/mnt/c/Users/kitya$ npm --version
8.19.4

```

Figure 6.1.7 JavaScript Version

```

kityann@Starfishhh:/mnt/c/Users/kitya$ node --version
v16.20.2

```

Figure 6.1.8 Node.js Version

Apart from that, fabric-samples repository from the GitHub provided by Hyperledger Foundation is cloned. This is to perform multiple tasks such as asset transfer, creation of network, and carry out network testing within organizations in a channel.

```
kityann@Starfishhh:~/fabric-samples$ ls
CHANGELOG.md          asset-transfer-ledger-queries  off_chain_data
CODEOWNERS            asset-transfer-private-data    test-application
CODE_OF_CONDUCT.md   asset-transfer-sbe             test-network
CONTRIBUTING.md     asset-transfer-secured-agreement test-network-k8s
LICENSE               auction-dutch                  test-network-nano-bash
MAINTAINERS.md       auction-simple                 token-erc-1155
README.md            bin                            token-erc-20
SECURITY.md          ci                             token-erc-721
asset-transfer-abac   full-stack-asset-transfer-guide token-sdk
asset-transfer-basic  hardware-security-module      token-utxo
asset-transfer-events high-throughput
kityann@Starfishhh:~/fabric-samples$ |
```

Figure 6.1.9 Fabric Samples Repository

6.2 Testing Setup and Result

Due to compatibility issues encountered during installation using the VM, especially with Docker versions, several adjustments were made to align the versions with the requirements of Hyperledger Fabric. Once compatibility was established, the sandbox environment was successfully configured, allowing for the testing of smart contracts. Despite limited resources, the system was able to process transactions securely and demonstrated a basic level of automation. However, further testing was hindered by the restricted number of nodes and services available under Kaleido's subscription model, limiting comprehensive performance testing.

6.3 Project Challenges

Several challenges emerged during the project, impacting the deployment and testing phases.

Deployment Issues with Hyperledger Fabric

One of the major challenges was the inability to successfully deploy the Pharmaceutical Drug Tracking system using Hyperledger Fabric. Despite numerous attempts and reconfigurations, the setup was unsuccessful, possibly due to outdated or misconfigured source code. This issue was exacerbated by the complexity of the deployment environment and the specific configurations required for the Fabric network.

Resource Limitations with Kaleido Platform

The Kaleido platform presented significant limitations, such as the need for a paid subscription to access more robust features and additional nodes. This restriction affected the project's ability to simulate a full-scale supply chain system, resulting in less comprehensive testing than originally planned. The limited resources also impacted the ability to showcase certain functionalities, such as private messaging, where other organizations could not view messages, they were not authorized to access.

Blockchain Technology Selection

Choosing the appropriate blockchain technology posed a challenge. The decision between Ethereum and Hyperledger Fabric required careful consideration of various factors, including security features, scalability, and integration capabilities. This comparison process was crucial for ensuring that the selected technology aligned with the project's goals and requirements.

Alternative Deployment Solutions

Finding an alternative to the Kaleido platform for deploying the environment was necessary due to the platform's constraints. Exploring other deployment solutions or platforms to better meet the project's needs became a priority, ensuring that the system could be effectively tested and demonstrated without the limitations imposed by Kaleido.

NFT Token Transfer Issues

Another challenge encountered was the inability to transfer NFT tokens to external wallets, such as MetaMask. This issue hindered the demonstration of token functionality and integration with external systems, impacting the overall scope of the project.

6.4 Objectives Evaluation

The project aimed to achieve several key objectives, and the results reflect significant progress in meeting these goals.

Objective 1: Integration of RPA into the supply chain

This objective focused on seamlessly integrating RPA into the supply chain security framework. Success in this area was achieved by invoking smart contracts using the Golang executable. By automating critical processes such as threat detection, vulnerability assessments, and access control verification, RPA contributed significantly to enhancing the security and efficiency of the supply chain. The use of Golang provided a robust and efficient way to implement the smart contracts, ensuring that the automation layer was both scalable and secure.

Objective 2: Boost overall efficiency in monitoring supply chain security

The second objective was to implement a secure SCM system using blockchain technology, specifically Hyperledger Fabric. This was successfully accomplished, with SCM running on top of the blockchain infrastructure, ensuring transparency, immutability, and enhanced security across the supply chain. Hyperledger Fabric provided the necessary tools and architecture to secure transactions and streamline processes, making the supply chain more efficient and secure.

Objective 3: Achieve scalability and adaptability in supply chain security

Scalability and adaptability were key goals for the project, and the integration of RPA within the blockchain-based supply chain security framework has proven effective. The system is designed to scale as the supply chain grows or as security needs evolve. The RPA-driven solution is flexible, allowing for adjustments to meet changing demands

and security requirements without needing a complete overhaul of the system. This adaptability ensures that the supply chain remains secure, regardless of its complexity or the number of nodes and transactions involved. By meeting both current and future security challenges, the project delivers a robust, scalable solution that aligns with the evolving needs of the supply chain.

6.5 Concluding Remark

In conclusion, this project successfully integrated RPA and blockchain technology into supply chain management to enhance automation, security, and transparency. Despite facing challenges, such as resource limitations and difficulties in fully deploying the Pharmaceutical Drug Tracking system, the project met its key objectives. The combination of smart contracts and blockchain demonstrated significant promise for improving supply chain operations. Moving forward, overcoming the constraints of the testing environment and addressing deployment issues would allow for a more comprehensive evaluation of the system's scalability and adaptability. The findings of this project indicate that the fusion of RPA and blockchain technologies offers a robust solution for modern supply chain management, paving the way for future developments in this field.

Chapter 7

7.1 Conclusion

Finally, this research delves into automation and blockchain technologies, with a particular focus on their application in SCM. The extensive investigation undertaken in the preceding chapters sheds light on different aspects of smart contracts and blockchain platforms, providing insights into their capabilities, authentication processes, and transaction flows.

The literature review in Chapter 2 dives further into smart contracts and popular blockchain platforms including Ethereum, Hyperledger Fabric, Quorum, and Corda. This review helped us grasp the structure of smart contracts, authentication and authorization mechanisms, and transaction procedures unique to each platform. This core information is provided as a starting point for future investigation and experimentation.

Chapter 3 presented the proposed technique and approach for developing a smart contract framework project, with a particular emphasis on using Hyperledger Fabric as the core blockchain infrastructure. The thorough system requirements, design diagrams, and transaction workflows outlined in this chapter provided the foundation for the project's succeeding phases. Chapter 4 detailed preliminary work, which included setting up the software environment. Chapter 5 provides hands-on with the sandbox environment created within the Kaleido platform., and also implementation using VM to invoke smart contracts and initializing Hyperledger Fabric. The practical studies on the creation of network, runtime environment, and creation of chaincode preceding the simulation of nodes. Chapter 6 evaluates the system and measures the performance.

Last of all, organizations may increase the transparency, traceability, and efficiency of their supply chains by employing smart contracts and blockchain platforms, leading to improved operational processes and better efficiency in the system. In conclusion, this research not only broadened our understanding of automation and blockchain technology but also demonstrated their disruptive potential in the field of SCM.

7.2 Recommendations

Extended Use Cases

To maximize the impact of my project, I opt to explore additional scenarios where RPA and blockchain technology can be applied within supply chain management (SCM). Beyond my current focus, consider integrating RPA and blockchain into other key areas such as procurement, inventory management, or logistics. For instance, in procurement, automated processes can handle supplier selection and contract management, while blockchain ensures transparent and secure contract execution. In inventory management, RPA can automate stock level monitoring and reorder processes, and blockchain can provide real-time tracking of inventory movement and provenance. Developing detailed case studies or simulations for these extended use cases can help demonstrate the benefits and potential of my integrated system in various supply chain contexts. This approach will not only showcase the versatility of the solution but also provide valuable insights into its practical applications.

User Experience and Usability

Improving user experience (UX) and usability is crucial for the adoption and effective use of your system. Start by designing intuitive and user-friendly interfaces for interacting with the RPA and blockchain solution. The goal is to create interfaces that simplify complex tasks and make it easier for users to perform their duties efficiently. For example, dashboards that provide clear visualizations of supply chain data and status updates can help users quickly assess and act on information. Additionally, develop comprehensive training and documentation materials to support users in understanding and leveraging the system's features. This should include user guides, tutorials, and troubleshooting tips. Effective training will empower users to maximize the benefits of the system and ensure smooth integration into their daily operations.

Compliance and Standards

Ensuring that the system complies with relevant regulations and industry standards is essential for its success and credibility. Start by verifying that the solution adheres to regulatory requirements such as GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act) for data protection and privacy. Additionally, align

the project with industry standards and best practices for RPA and blockchain in SCM. This includes ensuring the blockchain implementation follows standards for data integrity and security, and that RPA processes are designed to handle sensitive information responsibly. Compliance not only helps in meeting legal requirements but also builds trust with users and stakeholders, demonstrating that the system is both reliable and secure.

Collaboration and Feedback

Engaging with stakeholders and seeking feedback are crucial steps in refining the project and enhancing its relevance. Actively involve supply chain professionals, technology experts, and potential users in the development process. Their insights can help identify potential improvements and validate the approach. Consider organizing workshops, focus groups, or pilot programs to gather feedback and test your system in real-world scenarios. Additionally, explore opportunities for collaboration with industry partners or academic researchers. Collaborations can provide access to additional resources, expertise, and perspectives that can further enhance my project. By incorporating feedback and fostering partnerships, I can ensure that my solution meets the needs of its users and addresses any challenges effectively in the future.

REFERENCES

- [1] E. Puica, “How Is it a Benefit using Robotic Process Automation in Supply Chain Management?,” *Journal of Supply Chain and Customer Relationship Management*, pp. 1–11, Jan. 2022, doi: 10.5171/2022.221327.
- [2] N. Afriliana and A. Ramadhan, “The Trends and Roles of Robotic Process Automation Technology in Digital Transformation: A Literature review,” *ResearchGate*, Jul. 2022, doi: 10.33168/JSMS.2022.0303.
- [3] S. Rauch, “Top trends in RPA: Rise of intelligent Automation,” *Simplilearn.com*, Jul. 2023, [Online]. Available: <https://www.simplilearn.com/top-trends-in-rpa-article>
- [4] T. Sibalija, S. Jovanović, and J. S. Đurić, “ROBOTIC PROCESS AUTOMATION: OVERVIEW AND OPPORTUNITIES,” *ResearchGate*, May 2019, [Online]. Available: https://www.researchgate.net/publication/332970286_ROBOTIC_PROCESS_AUTOMATION_OVERVIEW_AND OPPORTUNITIES
- [5] S. Rauch, “Top trends in RPA: Rise of intelligent Automation,” *Simplilearn.com*, Jul. 2023, [Online]. Available: <https://www.simplilearn.com/top-trends-in-rpa-article>
- [6] “Monitoring and Facilitating Students Programming Skill Development using Robotic Process Automation (RPA) and Artificial Intelligence (AI)—A Case Study,” *IEEE Conference Publication | IEEE Xplore*, Dec. 26, 2022. <https://ieeexplore.ieee.org/document/10060772> (accessed Sep. 05, 2023).
- [7] M. Ammattikorkeakoulu, “Evolving digitisation: Chances and risks of robotic process automation and artificial intelligence for process optimisation within the supply chain,” *Theseus*, 2018. <https://www.theseus.fi/handle/10024/153503> (accessed Sep. 05, 2023).
- [8] “What is supply chain management? | IBM.” <https://www.ibm.com/topics/supply-chain-management> (accessed Sep. 02, 2023).

REFERENCES

- [9] J. Fernando, “Supply Chain Management (SCM): How it works and why it is important,” *Investopedia*, Jul. 2022, [Online]. Available: <https://www.investopedia.com/terms/s/scm.asp>
- [10] “What is Supply Chain Management? (SCM).” <https://www.oracle.com/scm/what-is-supply-chain-management/> (accessed Sep. 02, 2023).
- [11] S. Khan, R. K. Tailor, H. Uygun, and R. Gujrati, “Application of robotic process automation (RPA) for supply chain management, smart transportation and logistics,” *International Journal of Health Sciences (IJHS)*, pp. 11051–11063, Jun. 2022, doi: 10.53730/ijhs.v6ns3.8554.
- [12] G. Wright and S. Lewis, “supply chain security,” *ERP*, Apr. 2021, [Online]. Available: <https://www.techtarget.com/searcherp/definition/supply-chain-security>
- [13] D. Subramani, “Ethereum Architecture — Exploring node structures and consensus mechanisms,” Mar. 24, 2024. <https://www.linkedin.com/pulse/ethereum-architecture-exploring-node-structures-dhanaseelan-subramani-7yxdc/> (accessed Apr. 17, 2024).
- [14] M. GmbH, “Technical difference between Ethereum, Hyperledger Fabric and R3 Corda,” *Medium*, Nov. 16, 2018. Accessed: Apr. 17, 2024. [Online]. Available: <https://micobo.medium.com/technical-difference-between-ethereum-hyperledger-fabric-and-r3-corda-5a58d0a6e347>
- [15] L. Team, “A guide to quorum blockchain,” *LCX*, Jul. 22, 2022. <https://www.lcx.com/a-guide-to-quorum-blockchain/> (accessed Apr. 16, 2024).
- [16] R. Ecosystem, “Quorum Blockchain: A look at its use cases and pros and cons,” *Medium*, May 17, 2022. Accessed: Apr. 16, 2024. [Online]. Available: <https://medium.com/@rabit.ecosystem/quorum-blockchain-a-look-at-its-use-cases-and-pros-and-cons-e8a4255b0900>
- [17] M. Jeanne, “Security Assessment of Authentication and Authorization Mechanisms in Ethereum, Quorum, Hyperledger Fabric and Corda,” *Orange Cyberdefense*, Mar. 2019, Accessed: Sep. 06, 2023. [Online]. Available: https://www.epfl.ch/labs/dedis/wp-content/uploads/2020/01/report-2018_2-marie-jeanne-security-assessment.pdf

REFERENCES

- [18] “What is blockchain? | IBM.” <https://www.ibm.com/topics/blockchain> (accessed Apr. 17, 2024).
- [19] R. A. S, “What is Blockchain Technology? How Does Blockchain Work? [Updated],” *Simplilearn.com*, Oct. 18, 2023. <https://www.simplilearn.com/tutorials/blockchain-tutorial/blockchain-technology#:~:text=Blockchain%20technology%20is%20a%20structure,as%20a%20'digital%20ledger.'> (accessed Apr. 17, 2024).
- [20] Hindawi, “Table 1 | Smart Supply Chain Management using the blockchain and Smart Contract,” 2019. <https://cointelegraph.com/explained/how-blockchain-technology-is-used-in-supply-chain-management> (accessed Apr. 17, 2024).
- [21] Singh, “How blockchain technology is used in supply chain management?,” *Cointelegraph*, Nov. 26, 2022. <https://cointelegraph.com/explained/how-blockchain-technology-is-used-in-supply-chain-management> (accessed Sep. 07, 2023).
- [22] E. Androulaki *et al.*, “Hyperledger fabric,” *A Distributed Operating System for Permissioned Blockchains*, Apr. 2018, doi: 10.1145/3190508.3190538.
- [23] M. Renieri, “Ethereum Smart Contracts Optimization,” *University of Camerino*, 2020, [Online]. Available: <https://computerscience.unicam.it/marcantoni/tesi/Ethereum%20Smart%20Contracts%20Optimization.pdf>
- [24] S. Balasubramanian and I. S. Akila, “Blockchain implementation for agricultural food supply chain using hyperledger fabric,” *Journal of Intelligent & Fuzzy Systems*, vol. 43, no. 5, pp. 5387–5398, Sep. 2022, doi: 10.3233/jifs-211265.
- [25] I. Team, “Hyperledger Fabric: Definition, Example, Risks and 2.0 version,” *Investopedia*, Aug. 24, 2023. <https://www.investopedia.com/terms/h/hyperledger-fabric.asp#:~:text=What%20Is%20Hyperledger%20Fabric%3F,for%20use%20within%20private%20enterprises.>
- [26] “Introducing ChainCode: Hyperledger Fabric Smart Contracts.” <https://www.kaleido.io/blockchain-blog/introducing-chaincode-what-you-need-to-know#:~:text=Smart%20Contracts%20on%20Hyperledger%20Fabric,the%20automation%20of%20complex%20processes.>

REFERENCES

- [27] S. Sinha, S. Anand, and K. P. K, “Improving Smart Contract Transaction Performance in Hyperledger Fabric,” *Manipal Academy of Higher Education*, May 2021, doi: 10.1109/eti4.051663.2021.9619202.
- [28] H. Kang, X. Chang, J. Mišić, V. B. Mišić, Y. Yao, and Z. Chen, “Stochastic Modeling Approaches for Analyzing Blockchain: A survey,” *arXiv.org*, Sep. 13, 2020. <https://arxiv.org/abs/2009.05945>
- [29] A. Kumar, “About quorum - ABHISHEK KUMAR - Medium,” *Medium*, Aug. 02, 2018. [Online]. Available: <https://abhibvp003.medium.com/quorum-a52631b13018>
- [30] “What are smart contracts on blockchain? | IBM.” <https://www.ibm.com/topics/smart-contracts>
- [31] “Smart contracts and transactions | ConsenSys GoQuorum,” Aug. 28, 2023. <https://docs.goquorum.consenSys.io/develop/smart-contracts-transactions>
- [32] A. F. Mendi, T. Erol, and E. Şafak, “Generating a blockchain smart contract application framework,” *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 3, pp. 191–197, Jan. 2020, doi: 10.25046/aj050325.
- [33] R. Raab, “Towards Platform-Agnostic smart contracts,” 2021. doi: 10.34726/hss.2021.93502.
- [34] “Node.js — about Node.js®.” <https://nodejs.org/en/about>
- [35] Hyperledger Foundation, “Hyperledger Fabric,” *Hyperledger Foundation Project*, Dec. 01, 2023. <https://www.hyperledger.org/projects/fabric> (accessed Apr. 18, 2024).
- [36] “What is Kaleido? - Kaleido Docs.” <https://docs.kaleido.io/kaleido-platform/foundational-concepts/>
- [37] “Kaleido: Enterprise-Grade Blockchain & Digital Asset Platform.” <https://www.kaleido.io/>
- [38] GfG, “Go Programming Language (Introduction),” *GeeksforGeeks*, Apr. 24, 2023. <https://www.geeksforgeeks.org/go-programming-language-introduction/>
- [39] Y. Liu, Z. Zhou, Y. Yang, and Y. Ma, “Verifying the smart contracts of the port supply chain system based on probabilistic model checking,” *Systems*, vol. 10, no. 1, p. 19, Feb. 2022, doi: 10.3390/systems10010019.

REFERENCES

- [40] S. Al-Farsi, M. M. Rathore, and S. Bakiras, "Security of Blockchain-Based Supply Chain Management Systems: Challenges and opportunities," *Applied Sciences*, vol. 11, no. 12, p. 5585, Jun. 2021, doi: 10.3390/app11125585.
- [41] "Blockchain Technology for Secure Supply Chain Management: A Comprehensive Review," *IEEE Journals & Magazine | IEEE Xplore*, 2022. <https://ieeexplore.ieee.org/abstract/document/9841565>
- [42] M. D. Turjo, M. M. Khan, M. Kaur, and A. Zaguia, "Smart supply chain management using the blockchain and smart contract," *Scientific Programming*, vol. 2021, pp. 1–12, Sep. 2021, doi: 10.1155/2021/6092792.
- [43] M. Morley, "Top 5 use cases of Blockchain in the supply chain in 2021," *OpenText Blogs*, Feb. 12, 2020. <https://blogs.opentext.com/blockchain-in-the-supply-chain/>
- [44] "What is the SSDLC (Secure Software Development Life Cycle)?" <https://www.hackerone.com/knowledge-center/what-ssdlc-secure-software-development-life-cycle#:~:text=The%20Secure%20Software%20Development%20Life,developed%20with%20security%20in%20mind>
- [45] Aqua Security, "What is the Secure Software Development Lifecycle (SSDL)?" *Aqua*, Jan. 28, 2024. <https://www.aquasec.com/cloud-native-academy/supply-chain-security/secure-software-development-lifecycle-ssdl/>
- [46] H. Nguyen and L. Do, "The adoption of blockchain in food retail supply chain : Case: IBM Food Trust Blockchain and the food retail supply chain in Malta," 2018. [Online]. Available: <https://www.theseus.fi/handle/10024/158615>
- [47] G. FinTech, "Farm to Fork: Blockchain in the food ecosystem and IBM's Food Trust initiative," *Medium*, Dec. 03, 2018. [Online]. Available: <https://medium.com/georgetown-financial-technology-newsletter/farm-to-fork-blockchain-in-the-food-ecosystem-and-ibms-food-trust-initiative-c6cbdcf9c378>
- [48] "View of Application of Blockchain in Agri-Food Supply Chain." <https://biarjournal.com/index.php/bioex/article/view/233/266>
- [49] "IBM Supply Chain Intelligence Suite - Food Trust." <https://www.ibm.com/products/supply-chain-intelligence-suite/food-trust>
- [50] "IBM Food Trust product demo." <https://www.ibm.com/blockchain/resources/food-trust/demo/>

REFERENCES

- <https://www.hyperledger.org/case-studies/how-tradewaltz-is-using-hyperledger-fabric-to-create-the-future-of-global-trade>
- [57] TradeWaltz, “Trade Ecosystem | TradeWaltz,” *TradeWaltz | TradeWaltzTM Is a Trade Information Collaboration Platform Using Blockchain Technology. It Enables Safe, Smooth and Simple Trade and Brings Prosperity to People Around the World.*, Mar. 11, 2021. <https://www.tradewaltz.com/en/ecosystem/>
- [58] “About Walmart,” *About Walmart*. <https://corporate.walmart.com/about>
- [59] Hyperledger Foundation, “How Walmart brought unprecedented transparency to the food supply chain with Hyperledger Fabric,” *Hyperledger Foundation*, Aug. 01, 2023. <https://www.hyperledger.org/case-studies/walmart-case-study>
- [60] Simplilearn, “10 Practical applications of JavaScript & tips for a successful career,” *Simplilearn.com*, Aug. 13, 2024. <https://www.simplilearn.com/applications-of-javascript-article#:~:text=a%20JavaScript%20professional.-,What%20Is%20JavaScript%3F,buttons%2C%20control%20multimedia%2C%20etc.>
- [61] ““Docker Compose overview,”” *Docker Documentation*, Aug. 19, 2024. <https://docs.docker.com/compose/>
- [62] ““Overview of Docker Desktop,”” *Docker Documentation*, Aug. 13, 2024. <https://docs.docker.com/desktop/>
- [63] Craigloewen-Msft, “What is Windows Subsystem for Linux,” *Microsoft Learn*, Nov. 28, 2023. <https://learn.microsoft.com/en-us/windows/wsl/about>
- [64] “② Start your environment - Hyperledger FireFly.” https://hyperledger.github.io/firefly/latest/gettingstarted/setup_env/#a-firefly-stack

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3Y3	Study week no.: 2
Student Name & ID: Cheang Kit Yann 20AB03597	
Supervisor: Ts Dr Ooi Joo On	
Project Title: RPA in Secure Supply Chain Management	

1. WORK DONE

Met with the supervisor to discuss the Kaleido platform, including subscription details and its relevance to the project. Revised the FYP2 report based on supervisor feedback, specifically adding more detailed explanations of companies utilizing blockchain technology in supply chain management.

2. WORK TO BE DONE

Finalize the integration of Kaleido platform insights into the project. Continue updating the FYP2 report with additional details as required.

3. PROBLEMS ENCOUNTERED

There were no major issues that arose during this period.

4. SELF EVALUATION OF THE PROGRESS

Good progress in addressing supervisor feedback and enhancing the report with detailed explanations. Successful initial discussions about the Kaleido platform have set a clear path for future work.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3Y3	Study week no.: 5
Student Name & ID: Cheang Kit Yann 20AB03597	
Supervisor: Ts Dr Ooi Joo On	
Project Title: RPA in Secure Supply Chain Management	

1. WORK DONE

Inquired about implementing Hyperledger Fabric using Linux on Windows. Gathered information and resources related to this setup. Began initial implementation but encountered difficulties in configuration and integration.

2. WORK TO BE DONE

Resolve issues related to Hyperledger Fabric implementation on Linux within the Windows environment. Continue with further integration and testing of the Hyperledger Fabric setup.

3. PROBLEMS ENCOUNTERED

Difficulties encountered during the implementation of Hyperledger Fabric, particularly with configuring it on Linux through a Windows environment.

4. SELF EVALUATION OF THE PROGRESS

Progress is ongoing but has been hindered by technical difficulties. Further troubleshooting and research are needed to overcome the issues faced.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3Y3	Study week no.: 8
Student Name & ID: Cheang Kit Yann 20AB03597	
Supervisor: Ts Dr Ooi Joo On	
Project Title: RPA in Secure Supply Chain Management	

1. WORK DONE

Presented a partial showcase of the project to the supervisor for feedback. Requested a review of the FYP report, focusing particularly on the literature review section.

2. WORK TO BE DONE

Incorporate feedback from the supervisor into the project and report. Prepare for a more comprehensive presentation of the project.

3. PROBLEMS ENCOUNTERED

There were no major issues that arose during this period.

4. SELF EVALUATION OF THE PROGRESS

The project is progressing well with constructive feedback received. The partial showcase was beneficial for refining the project and ensuring alignment with expectations.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3Y3	Study week no.: 11
Student Name & ID: Cheang Kit Yann 20AB03597	
Supervisor: Ts Dr Ooi Joo On	
Project Title: RPA in Secure Supply Chain Management	

1. WORK DONE

Revised the FYP report based on supervisor's suggestions, including amendments to improve clarity and detail. Continued work on Hyperledger Fabric implementation, addressing issues related to using Linux on Windows.

2. WORK TO BE DONE

Finalize the implementation and testing of Hyperledger Fabric. Complete the remaining sections of the report and prepare for final submission.

3. PROBLEMS ENCOUNTERED

Ongoing issues with Hyperledger Fabric implementation on Linux within the Windows environment, requiring further troubleshooting.

4. SELF EVALUATION OF THE PROGRESS

Significant progress made with report revisions and addressing feedback. Implementation challenges are being actively worked on, with continued efforts to resolve the Hyperledger Fabric issues. Overall, the project is advancing well, with improvements in both documentation and technical setup.

Supervisor's signature

Student's signature

POSTER

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

Faculty of Information
and Technology

RPA in Secure Supply Chain Management

Introduction

This research explores RPA and blockchain technology in SCM. It delves into smart contracts and blockchain platforms, offering insights into capabilities, authentication, and transactions in supply chain.

Objectives

- To integrate the RPA technology into supply chain
- To boost overall efficiency in monitoring the supply chain security
- To achieve scalability and adaptability in supply chain security

Project Developer: Cheang Kit Yam
Project Supervisor: Ts Dr. Ooi Joo On
Project Moderator: Ts Deveendra Menon
a/I Narayanan Nair

PLAGIARISM CHECK RESULT

Robotic Process Automation in Secure Supply Chain Management

ORIGINALITY REPORT

8% SIMILARITY INDEX	5% INTERNET SOURCES	4% PUBLICATIONS	3% STUDENT PAPERS
-------------------------------	-------------------------------	---------------------------	-----------------------------

PRIMARY SOURCES

1	www.epfl.ch Internet Source	1%
2	buildmedia.readthedocs.org Internet Source	1%
3	eprints.utar.edu.my Internet Source	1%
4	Udit Agarwal, Vinay Rishiwal, Sudeep Tanwar, Rashmi Chaudhary, Gulshan Sharma, Pitshou N Bokoro, Ravi Sharma. "Blockchain Technology for Secure Supply Chain Management: A Comprehensive Review", IEEE Access, 2022 Publication	1%
5	Submitted to Universiti Tunku Abdul Rahman Student Paper	<1%
6	"Blockchain, IoT, and AI Technologies for Supply Chain Management", Springer Science and Business Media LLC, 2024 Publication	<1%

7	www.pitsolutions.ch Internet Source	<1 %
8	Arshad A. Dar, Faheem Ahmad Reegu, Gousiya Hussain. "Chapter 30 Comprehensive Analysis of Enterprise Blockchain: Hyperledger Fabric/Corda/Quorum: Three Different Distributed Leger Technologies for Business", Springer Science and Business Media LLC, 2024 Publication	<1 %
9	www.theseus.fi Internet Source	<1 %
10	repository.lib.ncsu.edu Internet Source	<1 %
11	Submitted to Australian Institute of Higher Education Student Paper	<1 %
12	Submitted to Universidade de Aveiro Student Paper	<1 %
13	www.frontiersin.org Internet Source	<1 %
14	Mohamed Torky, Aboul Ella Hassanein. "Integrating blockchain and the internet of things in precision agriculture: Analysis, opportunities, and challenges", Computers and Electronics in Agriculture, 2020	<1 %

PLAGIARISM CHECK RESULT

Publication		
15	Arya Kharche, Sanskar Badholia, Ram Krishna Upadhyay. "Implementation of blockchain technology in integrated IoT networks for constructing scalable ITS systems in India", Blockchain: Research and Applications, 2024 Publication	<1 %
16	andrew-coleman-fabric.readthedocs.io Internet Source	<1 %
17	www.mdpi.com Internet Source	<1 %
18	us.msi.com Internet Source	<1 %
19	Guangjie Lv, Caixia Song, Pengmin Xu, Zhiguo Qi, Heyu Song, Yi Liu. "Blockchain-Based Traceability for Agricultural Products: A Systematic Literature Review", Agriculture, 2023 Publication	<1 %
20	Submitted to Universiti Kebangsaan Malaysia Student Paper	<1 %
21	Yanji Piao, JongUk Kim, Usman Tariq, Manpyo Hong. "Polynomial-based key management for secure intra-group and inter-group communication", Computers & Mathematics with Applications, 2013 Publication	<1 %

PLAGIARISM CHECK RESULT

22	fastercapital.com Internet Source	<1%
23	www.zinio.com Internet Source	<1%
24	Submitted to Curtin University of Technology Student Paper	<1%
25	dl.uctm.edu Internet Source	<1%
26	Adel Ben Youssef, Pushan Kumar Dutta, Ruchi Doshi, Manohar Sajrani. "AI, Blockchain, and Metaverse in Hospitality and Tourism Industry 4.0 - Case Studies and Analysis", Routledge, 2024 Publication	<1%
27	Submitted to German University of Technology in Oman Student Paper	<1%
28	Submitted to Southern New Hampshire University - Continuing Education Student Paper	<1%
29	Submitted to University College London Student Paper	<1%
30	www.coursehero.com Internet Source	<1%

PLAGIARISM CHECK RESULT

31	Indranil Sarker, Bidisha Datta. "Re-designing the pension business processes for achieving technology-driven reforms through blockchain adoption: A proposed architecture", Technological Forecasting and Social Change, 2022 Publication	<1%
32	Submitted to University of Northampton Student Paper	<1%
33	www.getsmarter.com Internet Source	<1%
34	blogote.com Internet Source	<1%
35	www.experts-exchange.com Internet Source	<1%

Exclude quotes On

Exclude matches < 10 words

Exclude bibliography On

PLAGIARISM CHECK RESULT

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Cheang Kit Yann
ID Number(s)	20ACB03597
Programme / Course	FICT - DE
Title of Final Year Project	RPA in Secure Supply Chain Management

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u>8</u> % Similarity by source Internet Sources: <u>5</u> % Publications: <u>4</u> % Student Papers: <u>3</u> %	The percentage is within the accepted range.
Number of individual sources listed of more than 3% similarity: <u>0</u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Name: OOI JOO ON

Date: 11/09/2024

CHECKLIST FOR FYP2 THESIS SUBMISSION



UNIVERSITI TUNKU ABDUL RAHMAN

**FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)**

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB03597
Student Name	Cheang Kit Yann
Supervisor Name	Ts Dr. Ooi Joo On

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Title Page
✓	Signed Report Status Declaration Form
✓	Signed FYP Thesis Submission Form
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
✓	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
✓	Appendices (if applicable)
✓	Weekly Log
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
✓	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

Cheang

(Signature of Student)

Date: 27th August 2024