ERP Transaction Integrity Management Using Blockchain

MCS

Author

Junaid Ul Hassan

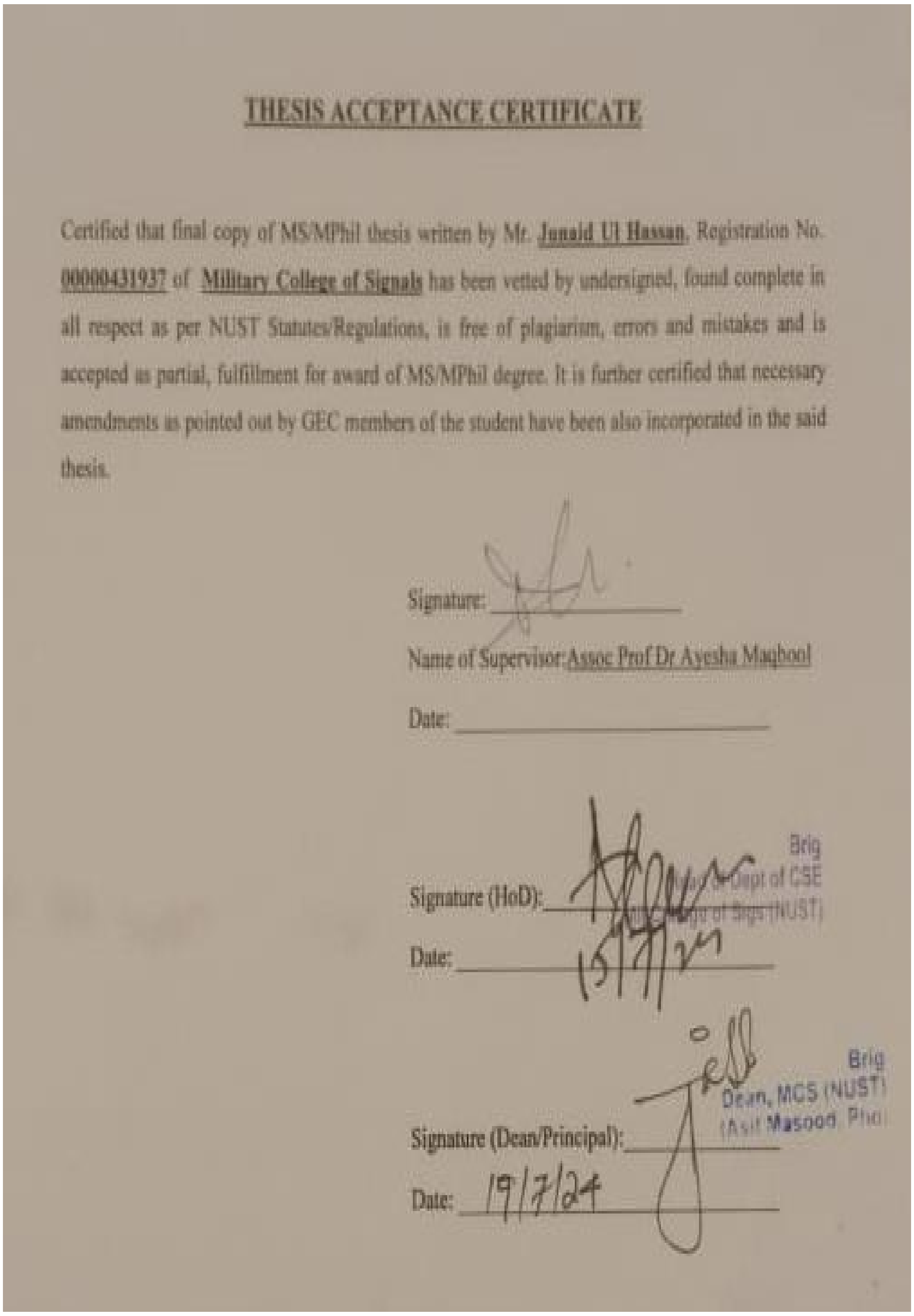
00000431937

Supervisor

Assoc Prof Dr. Ayesha Maqbool

A thesis submitted to the faculty of Computer Software Engineering Department, Military College of Signals, National University of Science and Technology, Islamabad, Pakistan in partial fulfillment of the requirements for the degree of MS in Computer Science

(June, 2024)



THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by Mr. **Junaid Ul Hassan**, Registration No. **00000431937** of **Military College of Signals** has been vetted by undersigned, found complete in all respect as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial, fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the student have been also incorporated in the said thesis.

Signature: [illegible]

Name of Supervisor: Assoc Prof Dr Ayesha Maqbool

Date: ____________________

Signature (HoD): [illegible]

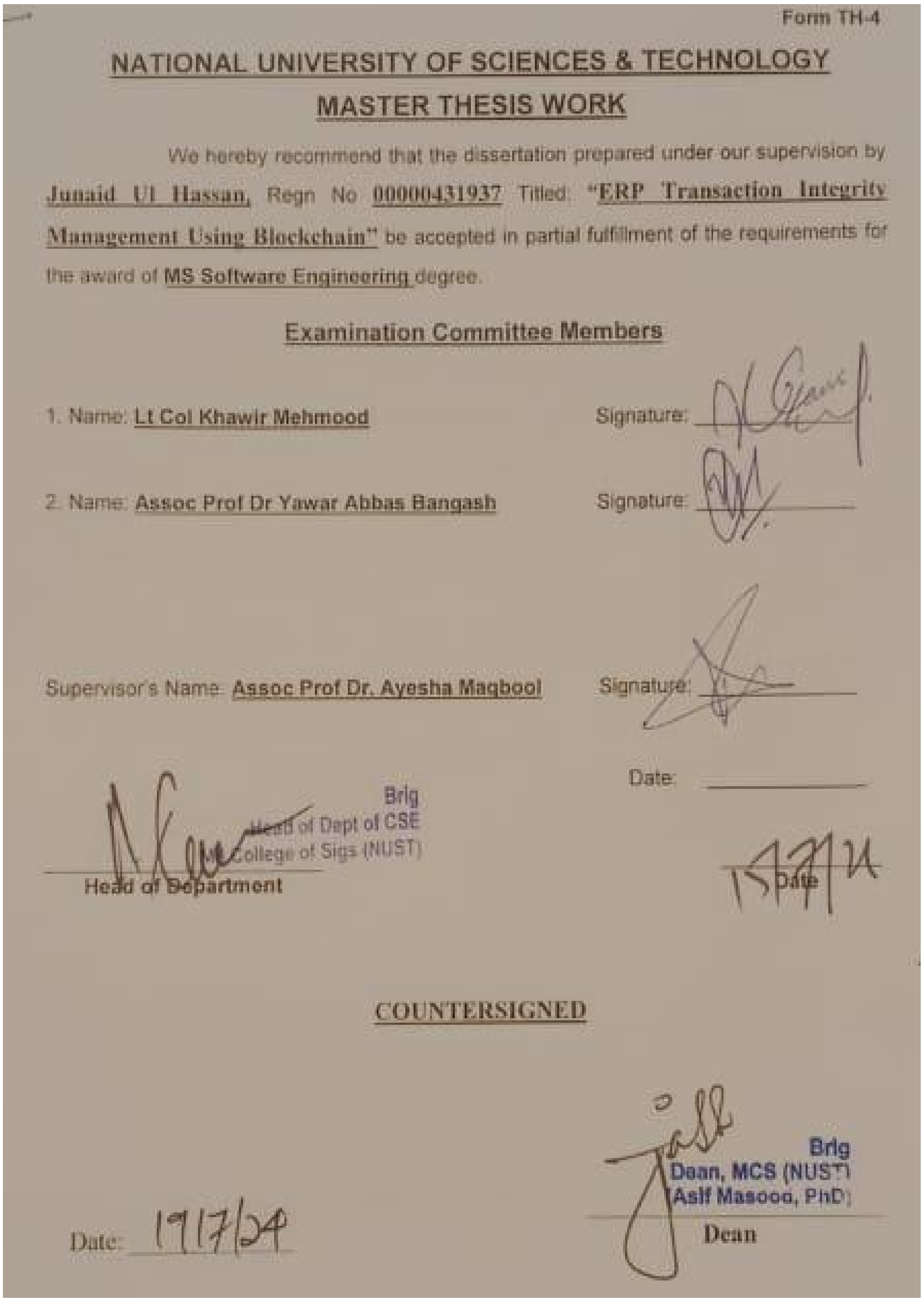
Brig
[illegible] Dept of CSE
[illegible] Sigs (NUST)

Date: 15/7/24

Signature (Dean/Principal): [illegible]

Brig
Dean, MCS (NUST)
(Asif Masood, PhD)

Date: 19/7/24



Form TH-4

NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY

MASTER THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by **Junaid Ul Hassan,** Regn No **00000431937** Titled: "**ERP Transaction Integrity Management Using Blockchain**" be accepted in partial fulfillment of the requirements for the award of **MS Software Engineering** degree.

Examination Committee Members

1. Name: **Lt Col Khawir Mehmood** Signature: [illegible]

2. Name: **Assoc Prof Dr Yawar Abbas Bangash** Signature: [illegible]

Supervisor's Name: **Assoc Prof Dr. Ayesha Maqbool** Signature: [illegible]

Date: ____________

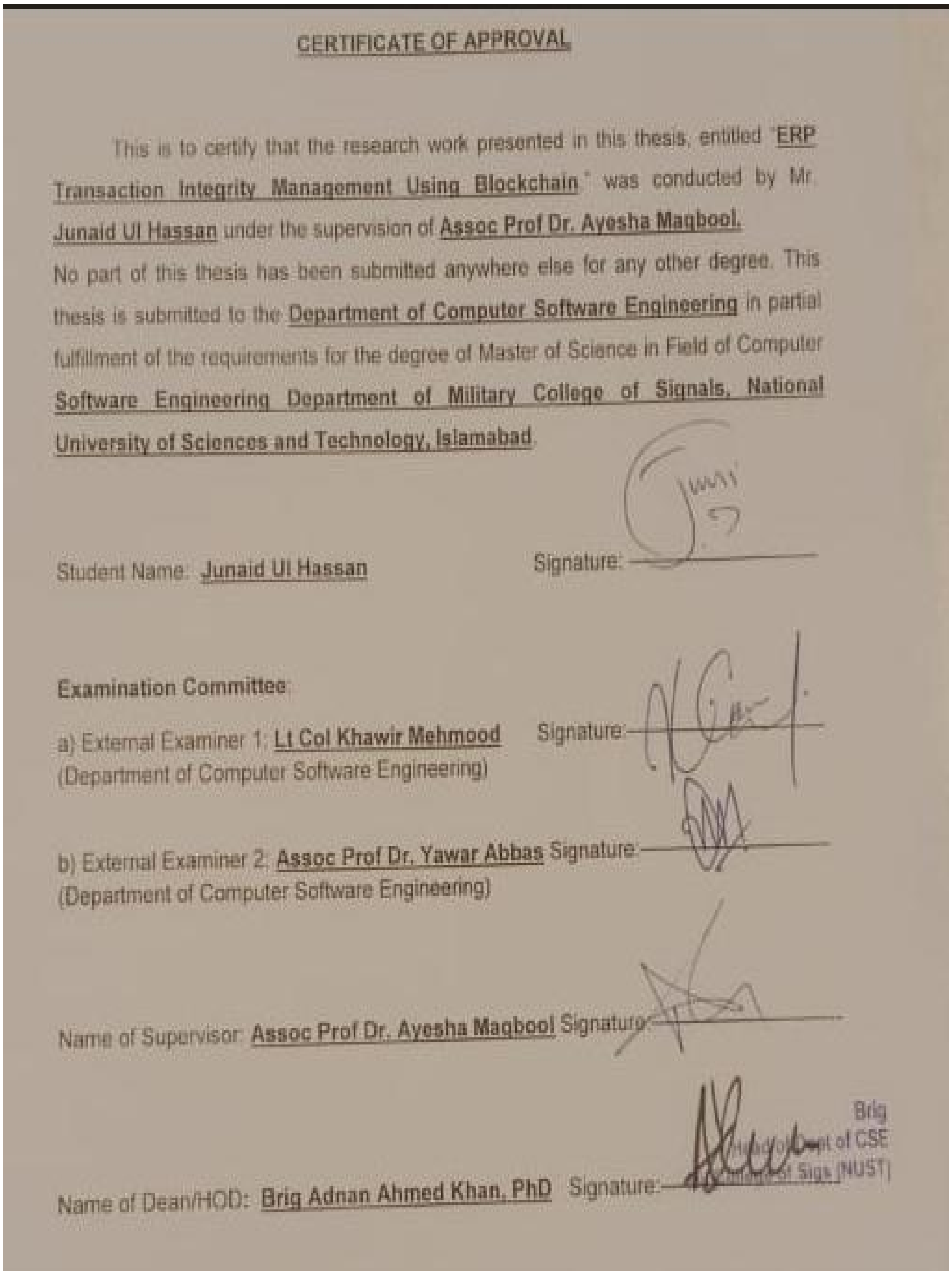
[illegible]
Brig
Head of Dept of CSE
MCollege of Sigs (NUST)
Head of Department

15/7/24
Date

COUNTERSIGNED

[illegible]
Brig
Dean, MCS (NUST)
(Asif Masood, PhD)
Dean

Date: 19/7/24



CERTIFICATE OF APPROVAL

This is to certify that the research work presented in this thesis, entitled "ERP Transaction Integrity Management Using Blockchain" was conducted by Mr. Junaid Ul Hassan under the supervision of Assoc Prof Dr. Ayesha Maqbool.
No part of this thesis has been submitted anywhere else for any other degree. This thesis is submitted to the Department of Computer Software Engineering in partial fulfillment of the requirements for the degree of Master of Science in Field of Computer Software Engineering Department of Military College of Signals, National University of Sciences and Technology, Islamabad.

Student Name: Junaid Ul Hassan Signature: [illegible]

Examination Committee:

a) External Examiner 1: Lt Col Khawir Mehmood Signature: [illegible]
(Department of Computer Software Engineering)

b) External Examiner 2: Assoc Prof Dr. Yawar Abbas Signature: [illegible]
(Department of Computer Software Engineering)

Name of Supervisor: Assoc Prof Dr. Ayesha Maqbool Signature: [illegible]

Name of Dean/HOD: Brig Adnan Ahmed Khan, PhD Signature: [illegible]

Brig
Head of Department of CSE
Military College of Sigs (NUST)

AUTHOR'S DECLARATION

I **Junaid Ul Hassan** hereby state that my MS thesis titled "**ERP Transaction Integrity Management Using Blockchain**" is my own work and has not been submitted previously by me for taking any degree from National University of Sciences and Technology, Islamabad or anywhere else in the country/ world.

At any time if my statement is found to be incorrect even after I graduate, the university has the right to withdraw my MS degree.

Student Signature: [illegible]

Name: **Junaid Ul Hassan**

Date: 24 July 2024

DEDICATION

Dedicated to my beloved family, whose boundless love, tireless support, and enduring encouragement have been the foundation of my academic journey. This thesis stands as a tribute to their selflessness and unwavering presence.

ACKNOWLEDGMENTS

"*Success is a collaborative effort, with the support of Allah, the wisdom of teachers, the prayers of family, and the encouragement of mentors*"

I am thankful to Allah Almighty for His unwavering support and the countless blessings He has bestowed upon me throughout my life. It is by His grace that I have achieved this milestone. Indeed, this research work could not have been completed without His support.

I am extremely grateful to my thesis supervisor, **Assoc Prof Dr. Ayesha Maqbool**, for her unwavering support and guidance throughout my thesis. Her knowledge, expertise, and dedication have been a source of inspiration to me, and I am grateful for the time and effort she invested in my success. Whenever I encountered any difficulties, she was always available to offer her assistance and provide me with insightful feedback.

I also extend my gratitude to my GEC members, **Lt Col Khawir Mehmood** and **Assoc Prof Dr. Yawar Abbas Bangash**, for their continuous availability and support throughout my degree, both in the coursework and thesis. Their knowledge and expertise have significantly contributed to the accomplishment of my task.

Junaid Ul Hassan

ABSTRACT

SecureERP is a centralized Oracle-based ERP system, and emerged as a single source of truth for many allied organizations within its network. Presently, traditional approach, such as, Service-Oriented-Architecture (SOA) and emails are used for data sharing among these organizations. Major ERP giants like Oracle, SAP, and Microsoft offer standardized functionality (providing pre-developed processes), while discouraging customization in the developed processes. However, there are use-cases, where organization has to deviate from standardized processes in order to meets its specific requirements. For example, in our use-case, extensive customization has been implemented in Oracle Enterprise Business Suite (^{e}BS) during the deployment of SecureERP. This customization has inadvertently introduced vulnerabilities, and raises concerns regarding transaction integrity and stakeholders' trust. The risk of single points of failure in the SecureERP system is effectively mitigated by establishing Disaster Recovery (DR) sites; however, ensuring transaction integrity is still a top concern. In this paper, we propose an architecture and framework, which seamlessly integrate Hyperledger Fabric (a Permissioned Blockchain) with SecureERP system to ensure transaction integrity and security. Our model uses a Raft-based consensus mechanism, which is more efficient compared to PoET, Kafka, ZKP (Zero-Knowledge Proof) and Proof of Work (PoW). Our proposed model is validated by conducting a Proof-of-Concept (POC) using relevant technologies.

INDEX TERMS. ERP Transactions Integrity, Hyperledger Fabric, Immutable Ledger, Integration of Blockchain and ERP

Table of Contents

ACKNOWLEDGEMENTS VIII

ABSTRACT IX

LIST OF FIGURES XIII

LIST OF TABLES XIV

LIST OF SYMBOLS, ABBREVATIONS AND ACRONYMS XVI

INTRODUCTION 1

1.1 Overview 2

1.2 SecureERP 3

1.3 Hyperledger Fabric Blockchain 4

1.4 Key Contributions 7

1.5 Problem Statement 7

1.6 Research Objectives 8

1.7 Relevance to National Needs 8

1.8 Area of Application 8

1.8.1 Advantages 9

1.9 Justification 9

1.10 Thesis Organization 10

LITERATURE REVIEW 12

2.1 Overview 13

2.2 Critical Analysis of Use-Cases 13

2.3 Synergy Bewteen Blockchain and ERP Systems 14

2.4 Related Work 16

2.5 Summary 18

RESEARCH METHODOLOGY 19

3.1 Overview 20

3.2 Design Conception 21

3.2.1 User Requirements Analysis..21

3.2.2 Research Ideation..22

3.2.3 Why Hyperledger Fabric..23

3.3 Proposed Architecture..25

3.4 Integration Scenarios..27

3.4.1 Scenario-1 : Future Use-Case..27

3.4.2 Scenario-2 : Present Use-Case..28

3.5 Txn Synchroniztion : Oracle Databas and Hyperledger Fabric..29

3.6 Design Analysis..30

3.6.1 Nodes Requirements..31

3.6.2 Participating Organizations..31

3.6.3 Orderer Service Configuration..32

3.6.4 Raft-based Consensus Mechanism..34

3.6.5 Peer Configuration and Block Size..35

3.6.6 Performance Consideration..36

3.7 Summary..37

IMPLEMENTATION OF PROPOSED MODEL..38

4.1 Overview..39

4.2 Environment..39

4.2.1 Infrastructure Setup..39

4.2.2 Devlopemnt of Application (Front-end)..40

4.2.3 Deployment of Hyperledger Fabric..41

4.2.4 REST API and Other Services Configuration..44

4.3 Summary..51

PROOF-OF-CONCEPT (RESULTS AND ANALYSIS)..52

5.1 Overview..53

5.2 Test-Case..54

5.2.1 Step-1 (Integration of SecureERP and Hyperledger Fabric)..54

5.2.2 Step-2 (Initiating Txn) 55

5.2.3 Step-3 (UpdatingTxn) 57

5.2.4 Step-4 (Deleting Txn) 59

5.2.5 Step-5 (Alerts Generation) 60

5.3 Summary 63

CONCLUSION AND FUTURE WORK 64

6.1 Overview 65

6.2 Feasibility of Proposed Model 65

6.3 Adoptability of Proposed Model 65

6.4 Challanges and Limitations 65

6.5 Future Work 67

96.6 Conclusion 68

REFERENCES 69

List of Figures

Figure 1.1: Illustration of Channel in Hyperledger Fabric Blockchain 6

Figure 1.2: Taxonomy of the Thesis 11

Figure 3.1: Block Diagram (Intergation of ERP and Hyperledger Fabric) 20

Figure 3. 2: Proposed Architecture and Framework 26

Figure 3. 3: Integration Scenario-1 28

Figure 3. 4: Itegration Scenario-2 29

Figure 3. 5: Raft-based Consensus Mechanism 35

Figure 4. 1: User Interface (Front-end Application) 40

Figure 4. 2: Running Docker Images 43

Figure 4. 3: Display Result on Front-end Application 49

Figure 4. 4: Ngrok - Graphical Display 51

Figure 5. 1: Integration of Ngrok with Front-end 55

Figure 5. 2: Integration of Ngrok with Front-end 55

Figure 5. 3: Item Creation in SecureERP : Front-end Application 56

Figure 5. 4: Transaction (Item Record) - Oracle Database (SecureERP System) 56

Figure 5. 5: Transaction (Item Record) - Blockchain (Hyperledger Fabric) 57

Figure 5. 6: Updating Transaction - SecureERP Application 58

Figure 5. 7: Updating Transaction - Oracle Database 58

Figure 5. 8: Updating Transaction -Blockchain---59

Figure 5. 9: Deleted Transaction - Hyperledger Fabric Blockchain 60

Figure 5. 10: Initial Status Before Alerts Generation 61

Figure 5. 11: Illegitimate Manipulation in Oracle Database 61

Figure 5. 12: Generated Alerts 62

List of Tables

Table 3. 1: Required Number of Nodes (Peers, Orderer etc) 31

Table 3. 2: Peer Nodes System Specifictaion 36

Table 4. 1: Installation of Pre-Requisite (Hyperledger Fabric) 42

Table 4. 2: GET HTTP Method (URI) Front-end Appliaction 45

Table 4. 3: Code Snippet (REST API) 46

Table 4. 4: Code Snippet (Chaincode) 47

List of Abbreviation

BPR	--------------------	Business Process Reengineering
BC	--------------------	Blockchain
DR	--------------------	Disaster Recovery
ERP	--------------------	Enterprise Resource Planning
^{e}BS	--------------------	Enterprise Business Suite
SCM	--------------------	Supply Chain Management
SOA	--------------------	Service-Oriented-Architecture
PoET	--------------------	Proof of Elapsed Time
ZKP	--------------------	Zero Knowledge Proof
POW	--------------------	Proof of Work
POC	--------------------	Proof of Concept
REST	--------------------	Representational State Transfer
API	--------------------	Application Programming Interface
Txn	--------------------	Transaction
URI	--------------------	Uniform Resource Identifier
JSON	--------------------	JavaScript Object Notation
URL	--------------------	Uniform Resource Locator
TPS	--------------------	Transaction Per Second
cURL	--------------------	Client URL
CFT	--------------------	Crash Fault Tolerance
PBFT	--------------------	Practical Byzantine Fault Tolerance

CHAPTER 1

INTRODUCTION

1.1 Overview

Enterprise Resource Planning (ERP) is the implementation of standard software modules for core business processes (i.e., Logistic, Financial and Human resources management) [3]. ERP systems integrate the procedures or processes of organization(s) into a single system having centralized database [2,3]. These systems are efficient and cost-effective, offering comprehensive solutions for all business processes within organizations [3,26,39]. Generally, organizations adopt ERP systems to deal with huge volume of operations or transactions, which are created from inside the organization [2]. Though single point failures emerge as a prominent problem in the ERP systems [3], however, maintaining the transaction's integrity in ERP like application remains a significant concern of enterprises.

Oracle, SAP and Microsoft are the major ERP proprietors across the globe. These proprietors provide their ERP systems with standardize functionalities. They expect and encourage enterprises to tailor their requirements according to already developed (built-in) processes in their ERP systems. In most cases, enterprises do so, and adhering to the principle of "Business Process Reengineering" (BPR). However, some times an organization(s) has to deviate from these built-in functionalities or processes available in these ERPs. In few cases, organizations require modification or customization in standard business processes to meet their requirements. This is where the problems regarding transaction integrity and complexity arise when alterations are made to the standardized functions or processes in any ERP systems.

This study is based on user-driven requirements to address transaction integrity concerns of an organization, where Oracle-based ERP (Oracle ^{e}BS) , named "SecureERP" is deployed . In-fact, SecureERP is a customized version of Oracle ^{e}BS, which means that extensive customization has been brought into Oracle ^{e}BS, in order to meet the organization needs.

Substantial research on the history, definition and general working of ERP systems are available and discussed by the authors of these papers [3,26,39]. Likewise, History and basic concepts regarding Blockchain technologies are well explained by Tehreem Aslam et al [3] and authors of [4,29]. Since, our study is based on a user-driven requirements and proposes a model that seamlessly integrates SecureERP and Hyperledger Fabric, with the aim to ensure transaction integrity in SecureERP application. Therefore, we escaped to rewrite general information about ERPs and Blockchain like technologies. Our discussion is restricted to SecureERP application and Hyperledger Fabric blockchain platform.

1.2 SecureERP

SecureERP is a centralized Oracle-based (^{e}BS) ERP system, deployed and operating within a single controlled organization. It is operational for the past **7-8 years**, and handles an average daily transaction volume ranging between **10000** and **15000**. As stated earlier, the major giant of ERP proprietors such as Oracle and SAP often encourage to use existing functionalities and processes for high performance, and to ensure compatibility, maintainability and security. The standard processes provided by the Oracle, go through rigorous testing phase, and offer high security in the underlying architecture. However, there are use-cases, where organization has to deviate from utilizing standard (already developed) processes. Like in SecureERP, where extensive customization has been brought into the Oracle ^{e}BS in order to meet the organization needs. Though this customization has fulfilled organization needs, however, it has introduced many inadvertent challenges that are being faced by the organization. Some of the Key challenges are as follow:-

a. **Transaction Integrity**: Customizing the standard processes and custom development led to inconsistencies, errors in the data handling and impacting the reliability of transactional records in SecureERP.

b. **Development Weaknesses**: Since ERP systems are complex, Extensive customization in ERP like system often requires complex development. This in turn increases the complexity, results in coding errors, logic flaws, or inefficiencies that ultimately weaken the overall stability and robustness of the system.

c. **Lack of Rigorous Testing and Quality Control**: Most ERP such as Oracle, SAP etc. are not free of cost and require subscription for the support. Therefore, quality and testing of these software are already ensured and guaranteed. However, when customization are made in these software, they may requires rigorous testing. In some cases, organizations may lack necessary resources or expertise of thorough testing and quality control, making the system vulnerable to unforeseen issues.

d. **Insider Threat**: Though this problem is common in all traditional databases, it is prevalent in the SecureERP, where administrators have full rights and privileges to access and can manipulate crucial data in the database.

1.3 Hyperledger Fabric Blockchain

Despite plenty of resources and literature available on Hyperledger Fabric, we want to provide a brief overview of its core concepts, and components in this study. This will help to revise basic understanding prior to discussing the design and implementation phase of our proposed model. Hyperledger Fabric is an open source private blockchain platform, initiative by Linux Foundation [10,41,42]. It is a Permissioned or private Blockchain platform, where only authorized organizations can make transactions. It supports cross-industry based distributed ledger [4,10,41]. In fact, there are several other private Blockchain platforms available under umbrella of Hyperledger, but Hyperledger Fabric stands out [10,41] due some key characteristics, such as, its modular architecture, provision of creating private channel, high transaction processing rate (TPR), comprehensive documentation and strong industry support [10,41] . Today, Fabric is one of the most widely adopted platform by the enterprises [10,41,42]. The key components of Hyperledger Fabric are briefly discussed in coming paragraphs:-

a. **Peer Node.** It is a node in the Hyperledger Fabric network that maintains a ledger and runs a chaincode (Smart Contract) as well. It has two roles, Endorser and Committer (the later does not have chaincode installed on it), and responsible to replicate and write block into the ledger [10,41,42]. The peer node plays an important role such as, endorsing and committing transactions in the Hyperledger Fabric network. Thus, the number of nodes participating in the network should be consider in such a way so that it can align with the TPR of ERP like application, which usually has very high TPR as compared to blockchain.

b. **Orderer Node.** It is also called an Orderer service. It is primarily responsible for ordering (Sequencing) of transactions in the block, creating blocks and forwarding these blocks to all the peers (Anchor Peer) in the blockchain network. The Orderer nodes (cluster) use b Raft- protocol for consensus mechanism (in our case) [10,41].

c. **Channel.** A channel is a communication path in the Hyperledger Fabric Network. Each channel has its own ledger. We can have multiple channels in a network [10, 41], as shown in the Figure 1.1. The organizations participating in a Hyperledger Fabric network can use one or more channels depending on their use case. However, the data or transactions are restricted to those organizations within the channel. In other words, only the organizations that are part of a specific channel can access the data or transactions on that channel. By design, channel provides confidentiality and privacy within a Hyperledger Fabric Network. This makes them suitable for scenarios, where only a specific set of participants need to see the shared information.

d. **Certificate Authority (CA).** CA is a service that is required to issue enrollment certificate (Private and Public Keys) to the authorized participants [10,41].

Hyperledger Fabric is modular [10,41], it allows participants to deploy their own custom CA. However, we used Fabric CA in the Proof-of-Concept.

e. **Chaincode.** It is a business logic (like Smart Contract in the public blockchain) that defines how to interact with Fabric network. We can invoke transactions using chaincode. It can be written in Go, Java, NodeJS (Typescript, JavaScript) [10,41]. We have written a chaincode in TypeScript, that performs Create, Read, Update, Delete (CRUD) operations on CouchDB (Database in Hyperledger Fabric) .

f. **Fabric Ledger/ Distributed Ledger.** It comprises State Database (CouchDB, LevelDB) and transaction logs (aka Blockchain) [10,41]. Blockchain is a data structure, where transactions or blocks are interlinked via hash. The first block is called genesis block and it does not link to any previous block, other blocks must have a hash reference to the previous block. Each Peer must have its own ledger. Hyperledger uses LevelDB by default, however, CouchDB is recommended instead of LevelDB for supporting complex JSON-based queries. We used CouchDB in our POC.

Channel - Hyperledger Fabric Blockchain

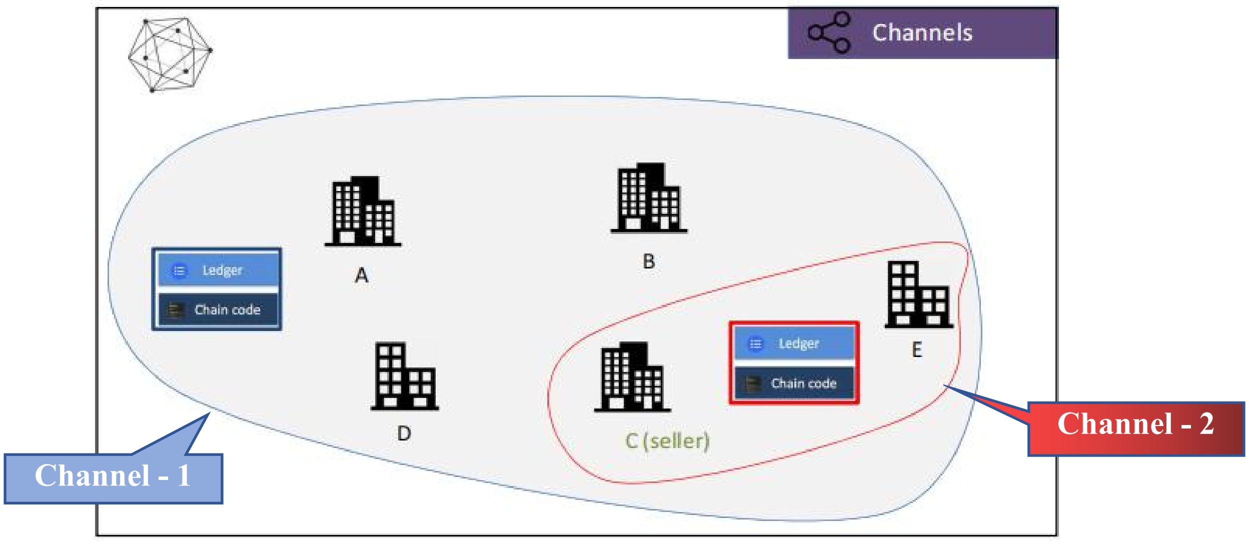


Figure 1.1 : Channel in Hyperledger Fabric Blockchain

Source: "http://ACloudFan.com"

1.4 Key Contributions

The key contributions of this research study are :-

a. We have developed Java based application using NetBeans 18 IDE. The aim is to represent key processes' functionality within SecureERP.

b. Designed a comprehensive architecture that seamlessly integrates Hyperledger Fabric with SecureERP application, providing a road-map for integrating Blockchain technology with any existing ERP systems.

c. Develop an environment or setup for conducting a Proof-of-Concept (POC). The POC validates our proposed model of integrating Hyperledger Fabric and SecureERP using Raft-based consensus mechanism.

d. Developed chaincode in TypeScript for CRUD operations at blockchain level.

e. REST APIs are developed that act as a bridge between SecureERP and Fabric Network.

f. Integration of Hyperledger Explorer with Fabric Network for transparency and visibility of blockchain transactions and monitoring.

1.5 Problem Statement

SecureERP is a centralized and customized version of Oracle-based ERP system. To meet the organization's needs, extensive customization has been implemented into the Oracle Enterprise Business Suite (^{e}BS). This customization has introduced certain problems and challenges that make data and transaction susceptible to unwanted changes. The state and transactional data in the SecureERP are vulnerable to insiders (database and application administrators) attacks. Furthermore, the lack of adherence to proper testing and quality control measures has worsen these challenges. These vulnerabilities have raised concerns regarding transaction integrity and stakeholders' trust. To address these challenges, there is a need to propose a Blockchain-based framework that seamlessly integrates with SecureERP system. The proposed architecture must be modular, scalable, operable within an ERP environment, and above all it must have high transaction processing rate (TPR). Finally, the overall purpose is, to ensure transaction integrity within the SecureERP system and to maintain stakeholders' trust.

1.6 Research Objectives

The main objectives of this study are:-

a. Critical analysis of the issues or problems, that affect transaction integrity in SecureERP application
b. Propose a framework that integrates SecureERP with Blockchian, the purpose to ensure the transaction integrity and security in the SecureERP system.
c. What challenges an enterprises may face while integrating existing ERP systems with Blockchian technology.

1.7 Relevance to National needs

Securing transaction and making it immutable using Blockchain technologies could be seen as directly addressing a critical need for Cyber Security, and to control insiders attacks in enterprises.

1.8 Area of Application

The adoption to our proposed model, offers a robust solution to safeguard transaction integrity within an ERP like systems, such as the SecureERP application. Generally, our model or technique can aid following organizations from an illegitimate transaction or fraudulent activities:-

a. Supply Chain Management
b. Healthcare specifically in Hospitals Management System

c. Education, Excise & Taxation, and many government sectors which support XA execution

1.8.1 Advantages

Key advantages are:-

a. By ensuring the integrity of transactions, we can build greater trust with users or stakeholders, which can be particularly important in sensitive applications such as SecureERP application.

b. Using Hyperledger Fabric provides a high level of data security by allowing participants to create a private channel. So, unlike other private blockchain platforms such as Sawtooth etc, Hyperledger Fabric allows organization to maintain secrecy and confidentiality.

c. Integrating Blockchain such as Hyperledger Fabric with ERP like application can mitigate the risk of single failure.

d. Blockchain improves the data security by providing an immutable and tamper-proof transaction logs, which makes forensic analysis much easier and helps in identifying any unauthorized changes or tampering of data.

1.9 Justification for the Selection of the Topic

This research is driven by a clear objective to address user-driven requirements for transaction integrity within the SecureERP application. The significance of this choice is obvious in several key aspects as discussed below:-

a. **User-Centric Approach**. Since the Organization prioritizes transactional integrity. Therefore, we aim to provide solution that fulfills its needs. Transaction integrity is a critical for organization, because it affects the accuracy, reliability, and trustworthiness of the system.

b. **Mitigating Risks**. Addressing the integrity concerns of organization, helps safeguard against financial errors, data discrepancies, operational inefficiencies, and compliance violations.

c. **Organizational Efficiency**. A strong transaction integrity framework complements the performance of the ERP application with the aid of minimizing mistakes and maintaining data consistency.

d. **Stakeholder Trust**. Addressing the transaction integrity concerns of organization, this research contributes to maintains stakeholder's trust. Meeting user-driven requirements for transaction integrity reinforces stakeholder confidence and strengthens relationships.

e. In summary, our research is **focused on user-driven requirements**. The aim is to provide solutions that fulfill the organizational needs

1.10 Thesis Organization

The study is organized in the following chapters:-

- **Chapter 1**: A brief introduction is given. Research objectives are listed. Relevance to National need is highlighted followed by an area of application, its advantages and justification for selection of the topic is elaborated.
- **Chapter 2**: Describes the related works and literature review carried out on Blockchain and ERP integration. A comparison is drawn to observe existing work by various researchers.
- **Chapter 3 & 4**: Discuss the overall research methodology including, overview of proposed model, design Ideation and its implementation strategy.
- **Chapter 5**: This Chapter presents implementation of proposed model with Proof-of-Concept, conducted to validate the results and objective achieved by our proposed model.

- **Chapter 6**: This Chapter sums up the research with conclusion drawn and provides direction for future work
- **Chapter 7**: References are listed

Taxonomy of the Thesis

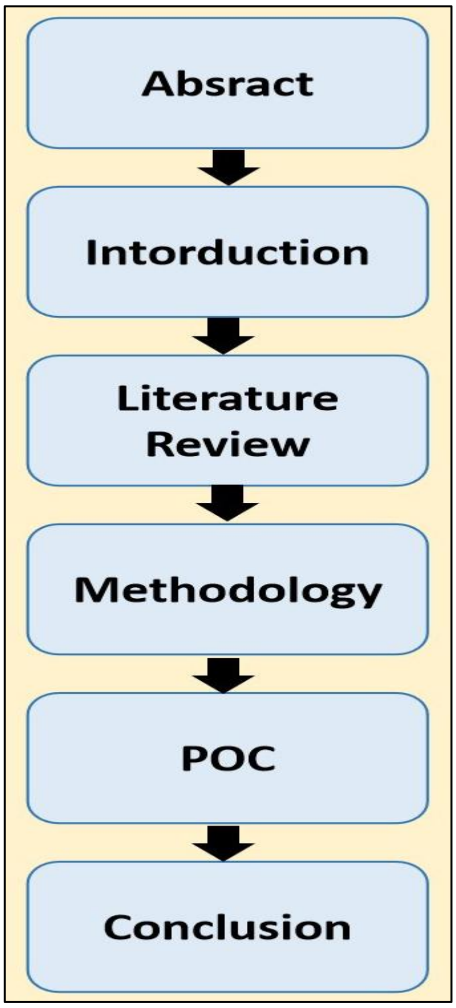


Figure 1.2: Taxonomy of the Thesis

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

In Chapter 2, literature review and related work are discussed. It explains about the existing work carried out by the former researchers along with their findings and results. Currently, there is an increasing interest shown by enterprises in an integrating ERP systems with the Blockchain technology. The reasons are, to enhance transparency, achieving temper-proof and immutable ledger, and to avoid single failure chances in the ERP like systems. The goal of this section is to analyze the related and previous research work done on integrating ERP systems with Blockchain platform. We have summarized existing literature review in three categories i.e. Analysis of use-case, Synergy between Blockchain Technology and ERP, and related work.

2.2 Critical Analysis of Use-Case

According to Deloitte's Global Blockchain Survey 2021, "***Blockchain offers a compelling business case***". However, to the best of our knowledge, it is not always possible to stack blockchain into any existing IT infrastructure. Therefore, it is necessary for enterprises to thoroughly analyze use-case before bringing blockchain into their IT environment. Moreover, the report published by International Data Corporation (IDC) [48], described that blockchain projects have very high failure rate because of complexity and Hardware cost involved in these projects. Few Former researchers have also emphasized to analyse use-case prior to implementing Blockchian. For example, in the paper [1], Chowdhury, Alan Colman et al (2018), conducted a critical analysis of both traditional database systems and Blockchain technologies. The author [1] identifies use-cases for Blockchain implementation and provided a "Decision Tree". This will help decision makers, whether to use blockchain or database in their organizations. Similarly, A R Komala et al. (2020) in their paper [8] highlighted the possibility of utilizing Blockchain as a medium for operating ERP like systems. Moreover, Lahlou Imane et al (2023) in their paper [9] provided per-implementation guide for blockchain technology. This guide helps decision-maker to study the feasibility of Blockchain and ERP integration [9]. Arman Raj, Vandana Sharma et al (2023) [13], used Multichain-based Blockchain technology in order to ensure security and transparency in the financial transactions. This study proposes a decentralized P2P Network or framework for end-to-end transactions with user legitimacy. It also discussed important impact of "Blockchain on market like Equity Market, Wealth Management, Payment and Remittances, Commercial Lending, and Insurance. This study suggests that decentralized P2P Network/ framework can mitigate the risk of single point failure, the authors recommend to explore this framework beyond financial transactions [13]. Similarly, Karthik H P et al (2023) exercise the case study of business-to-business connections in supply chain integration, "drawing data from a focus group comprising CEOs, Business Managers, and IT specialists from 38 companies across various sectors" [14]. The study focuses on discovering gap between Blockchain integration in supply chains [14]. In addition, the paper [37] carried out a systematic review of 29 papers on Blockchain challenges in Supply Chain Management, with a purpose to aware top management regarding obstacles in Blockchain technology and advising against blind adoption [37].

2.3 Synergy Between Blockchain and ERP Systems

It is worth noting that ERP and Blockchain are two different technologies, both serving different purposes. Numerous research studies have been conducted by previous researchers on the potential of blockchain in respect to ERP systems. Such as, in the paper [2], Tinal Parikh et al (2018), discussed the strength of Blockchain technology to transform the existing ERP systems by integrating with Blockchain for enhancing security, improving and streamline the existing business processes [2]. Similarly, LAHLOU Imane et al (2022) author of [4], highlighted that due to advancements in the technology (such as Blockchain,Web3, IoT etc.) provides opportunities and benefits for supply chain management (SCM) to integrate with Blockchain for better performance and security [4]. Another study conducted by Alessio Faccia, and Pythagoras Petratos et al (2021) in their paper [7], contribute theoretical aspects to assess Blockchain integration with ERP and Accounting Information system (AIS). However, their research is limited to theoretical approach, the authors suggesting empirical approach to test the potential of blockchain [7]. Additionally, the authors of the papers [20,29,36] echo same opinion as the other sources cited ([2], [4], [7]) to explore the potential of Blockchain and SCM. A R Komala et al (2020) the authors of [22], discussed the feasibility of Blockchain for the operational ERP. The study adopted descriptive qualitative approach, and Stellar Blockchain platform is used to test the transmission rate [22]. Their proposed model may not be feasible for enterprises where data privacy is preferred over data transparency, since Stellar is a public blockchain platform.

Glory Ugochi Ebirim et al. (2024) gives a critical review of ERP implementation within Multinational Corporations (MNCs), conveying current trends, challenges, and future directions. It also emphasized to explore the power of emerging technologies like Blockchain, Industry 4.0 etc, while adopting ERP in MNCs. The authors stresses on future research regarding impacts of Industry 4.0 and Blockchain technologies on ERP systems [28].

UDIT AGARWAL et al (2022) gives future directions and research areas on the role of Blockchain (BC) in Supply Chain Management (SCM). The authors believe that " ***risks and challenges such as high storage costs, scalability etc can be addressed, when BC is used with advanced tools like 5G, IoT, machine learning, and Cloud computing***" [32]. The paper highlighted the increasing interest in BC based research and leave open research questions for future work [32]. Thomas Kitsantas et al (2022) [34], adopted a theoretical approach on integrating of Blockchain Technology (BT) with ERP. The author acknowledges limitations, "such as the theoretical nature of proposed deployments and the early stage of BT [34]. The study suggested to examine BT's cyber-security related features, interoperability with legacy systems, and effects on Industry 4.0 components by doing some practical research [34]. ***The outcomes of these studies outline that Blockchain can complement ERP systems rather to replace them*** [20,29].

2.4 Related Work

In this section, we'll look at some important previous research work that explore how blockchain can be used with existing ERP systems. Understanding of these papers will help us to identify challenges while integrating both the technologies. By reviewing the related work, we'll get a clarity of what has been done so far and what still to be done?

In the paper [3], Tehreem Aslam, Ayesha Maqbool et al, proposed an architecture that integrates Blockchain with the BlockERP to ensure security, transaction integrity and to address single point failure problem in ERP like systems. The authors validate their proposed model using Hyperledger Sawtooth (Private Blockchain Platform) and Proof of Elapsed Time (PoET) protocol for consensus mechanism. PoET is more efficient than Proof-of-Work (PoW). However, PoET also follows a probabilistic approach, where each validtor node gets a random time to create block(s) in the blockchain. In our point of view, an enterprises which operate in a controlled environment, as in our case SecureERP, require a simple and straightforward consensus protocol for creation of blocks in the blockchain. Adopting Raft-based consensus mechanism and Hyperledger Fabric instead of Sawtooth and PoET will not only increase efficiency but also align to their operational environment as well. We will discuss Raft protocol consensus mechanism in details in the design and implementation section (Understanding PoET Protocol, n.d.) [3, 6,10].

Similarly, [5] presents a empirical case study of integrating Hyperledger Fabric and Odoo ERP. This paper [5] discussed two integration models: " (1) Common Model Integration Architecture for fresh ERPs and (2) Existing ERP Integration Architecture for established ones" [5]. However, the paper [5] implemented the Common Model, which transforms data into a common format before sending to Odoo ERP and Blockchain, bu it gives a solid foundation for our research work by transforming the second model into pragmatic form.

Another study [6] carried out by Kok Yong Chan et al (2019), uses exiting literature and developed a Blockchain-based framework for agri-food sector in Malaysia for traceability and transparency. They proposed conceptual framework named “Prochain” having key participants like farmers, processors, distributors, retailers, and consumers etc. This study prefers to adopt Hyperledger Fabric over Sawtooth for privacy and confidentiality [6]. In addition, A R Komala et al (2020) the authors of [22], discussed the feasibility of Blockchain for operational ERP. The study adopted descriptive qualitative approach and Stellar Blockchain platform is used to test the transmission rate [22]. It is worth mentioning that Stellar is a public Blockchain platform and may not feasible for those enterprises where data privacy is preferred over transparency.

Moreover, MUHAMMAD IMRAN SARWAR et al (2021) author of [11] proposed a hybrid solution to ensure data integrity in “Accounting Information System (AIS)” against deliberate alterations from the database. The authors [11] adopted a conceptual approach of using Data Vaults, which serve as a storage for ensuring data integrity in AIS. A cryptographic SHA256 are applied to the data in the Data Vaults. This paper [11], demonstrated a simple, but robust version of custom Blockchain to assist organization in storing financial and accounting data. Their study stress the need to “verify system capacity and scalability, particularly regarding emerging threats like Quantum computing ” [11]. Enterprises having centralized database, and looking for limited functionality of blockchain like immutable or data integrity may utilized the model presented in the paper [11].

Similarly, in the paper [12], the authors explore the capabilities of Orion Blockchain (IBM) database for immutability, transparency among multi-party data access control system. This study compares Orion Blockchain database with other Blockchain databases such as Azure SQL, QLDB etc, and other Blockchain platforms (Hyperledger Fabric, Sawtooth etc). The authors pointed the dominance (Performance, Simplicity and Costs etc) of the Orion in future application [12]. However, it is worth noting that Orion does not support Smart contract like functionality of core Blockchain technology. Ultimately, the choice between adopting Orion and other Blockchain platforms depends on the use-case and the trade-off involves.

2.5 Summary

The studies reviewed, cover various types of literature based on use-cases, potential of blockchain technology and its impact on future ERP like system. Various researchers emphasized on proper analyzing of use-case before inheriting blockchain capabilities into their existing IT infrastructure. In addition, few papers [3,5] in specific, done an excellent work by adopting an empirical approach and provide a solid ground on the role of Blockchain in ERP like application. To summarize this literature review, we can conclude that Blockchain complements ERP system, however, it may not replace ERP systems.

In our proposed model, we use Hyperledger Fabric instead of Sawtooth and preferred Raft consensus protocol over PoET. We evaluate that Hyperledger Fabric, combined with the Raft consensus mechanism, would give better results over Sawtooth and PoET, which is further discussed in the “Methodology” section .

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Overview

This chapter describes the design and implementation phases of our research. Since, this research is based on user-driven requirements to ensure transactions integrity in Oracle-Based ERP system, aka SecureERP. In the context of problem statement, we examined existing related work and literature review. We then continued our research with the idea of integrating existing ERP with the blockchain as discussed in the paper [5]. Similarly, the work done by Tehreem Aslam et al (2021) [3], provides an excellent groundwork for our research. In the design phase, we proposed a model as depicted in Figure-3.2, which conceptualizes the integration of Hyperledger Fabric with the SecureERP system. In order to validate our model, we conducted a Proof-of-Concept (POC) based on proposed architecture (Figure-3.2). Since, Figure-3.2 is comprehensive and covers complete transactional flow in both the systems, therefore, to increase readability and understandability of our design, we have drawn a block diagram as well, as shown in Figure-3.1, this block diagram provides a high level perspective of our work.

Block Diagram - Integration of SecureERP and Hyperledger Fabric Blockchain

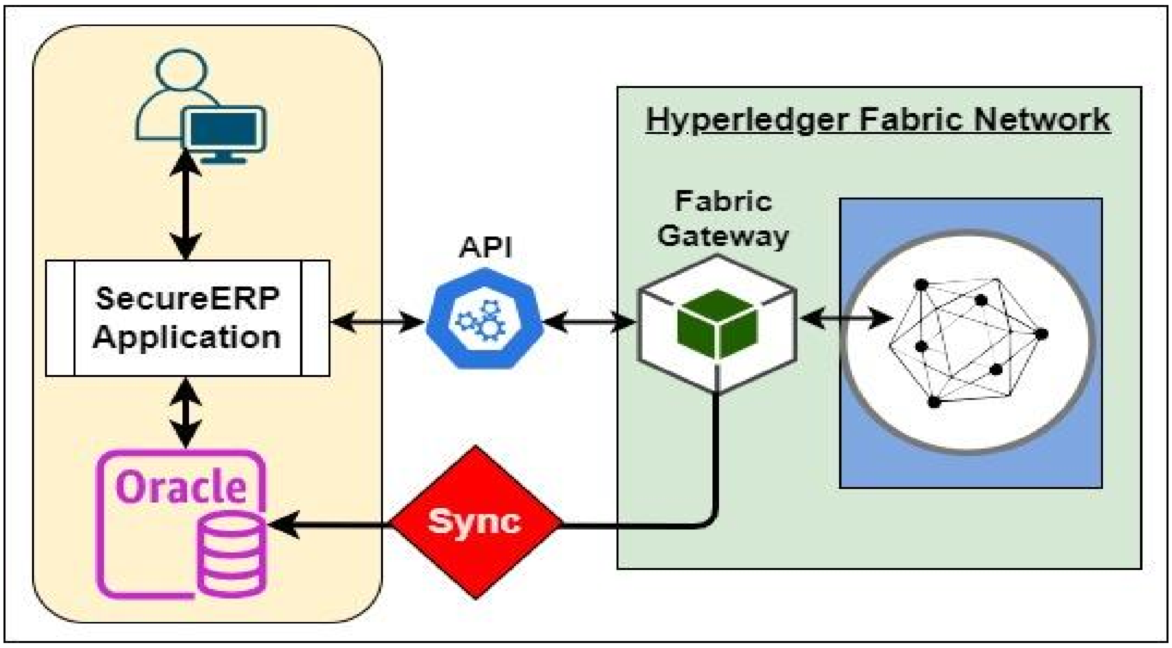


Figure 3.1 : Block Diagram (SecureERP and Hyperledger Fabric Integration)

Fig. 3.1, represents the abstract view of integrating SecureERP with Hyperledger Fabric

3.2 Design

In this phase a series of steps were taken to translate the requirements (as highlighted in the problem statement) into a robust architectural framework. In the coming sections, we delve into the details of these tasks or steps. Based on the analysis, a model and framework have been proposed by the authors that seamlessly integrates SecureERP with Hyperledger Fabric blockchain platform. The proposed model addresses the concerns of an organization regarding transaction integrity and security, and serves as a blueprint for the implementation phase. The proposed model is validated by conducting a POC using relevant technologies. Now in coming paras, we will discuss the steps or measures that were taken in the Design phase.

3.2.1 User Requirement Analysis

SecureERP is a powerful ERP application, acting as a single source of truth and central data repository for many allied organizations within its network. SecureERP is a customized version of Oracle ERP (^{e}BS). Although, customization is done to meet the organization's needs, but it has introduced few critical challenges, that need to be addressed. We have identified key requirements of the organization through consultation, feedback sessions and examined existing functionality of key processes in the SecureERP. The most important among these identified requirements are:-

- Transactions in SecureERP system are susceptible to illegitimate changes and manipulations. Thus, integrity of transactions must be ensured at any cost.
- In traditional databases, administrators can manipulate data, even if audit are enabled, since enabling audit can be reactive not proactive. Similarly, in our case, administrators of SecureERP have been granted with excessive privileges that have serious consequences if misused. Therefore, it is important to address vulnerabilities related to broad access and privileges with these admins.

- Presently, SecureERP lacks distributed transaction support. It is relying on traditional services, such as, Service Oriented Architecture (SOA) and email for data sharing among organizations. Thus, there is a need to have an immutable, auditable ledger of transaction logs that can be referenced for audit purposes. Overall, the aim is to keep record of transactions and access logs, which cannot be fraudulently manipulated.

3.2.2 Research Ideation

After understanding the key requirements, we adopted a systematic approach to find a solution that can address the challenges identified within the existing SecureERP system. Firstly, we carried out detailed analysis of those issues that raise concerns regarding transaction integrity and security. The prominent issues identified are:-

- **Insiders Threat.** The state and transactional data in the SecureERP are vulnerable to database and application administrators. At database level, though audits are enabled that keep track of activities in the database system. However, these audits do not restrict insiders from an unwanted or illegitimate activities. So, the access privileges with these administrators can have serious consequences on the system if misused.

- Customization done in the SecureERP has not only increased the complexity, but also made the system difficult for the administrators to operate and maintain. This customization with lack of adherence to testing and quality control measures, caused performance related issues in the SecureERP system as well.

It is worth mentioning that, in the SecureERP system, data confidentiality is more critical and usually preferred over data transparency. After studying existing literature and related work, particularly the idea presented by Tehreem Aslam et al (2021) in their paper [3], and Abdelhak Belhi et al (2021) in [5], ***We concluded that integrating Hyperledger Fabric with***

SecureERP will address all these challenges, such as transaction integrity, confidentiality, immutability, insiders' attack and point of single failure. The rationale behind our decision to choose Hyperledger Fabric in so many private Blockchain platforms is discussed in the next section. To validate our design or model, we plan to conduct POC that will demonstrate integration of Hyperledger Fabric blockchain with SecureERP system with the aim to ensure transaction integrity in the SecureERP system.

3.2.3 Why Hyperledger Fabric?

According to Hyperledger [10] " Hyperledger Fabric is designed to meet diverse industry needs. Additionally, it offers a unique approach to consensus that facilitates scalable performance while maintaining privacy " [10]. Our approach of integrating ERP applications with Blockchain is different from that presented by Tehreem Aslam et al in their paper [3]. We used Hyperledger Fabric with Raft-based consensus mechanism instead of Sawtooth and PoET. The key reasons behind choosing Hyperledger Fabric are:-

- **Modular Architecture**. Like Sawtooth, Hyperledger Fabric is a Permissioned Blockchain Platform, known for its modular architecture that easily integrate with other systems, and allows plug-and-play functionality (i.e. Language based SDK, Certificate authority, writing chaincode in different languages etc)" (Understanding the Hyperledger Fabric, n.d.) [10]. This modular design architecture will help enterprises to build network according to their desired use-case(s).

- **Efficient Consensus Mechanism**. Unlike PoET (which is a probabilistic), Raft uses deterministic, straightforward and efficient consensus mechanism to create block in the network. It is important to highlight that Raft can be configured with both CFT (Crash Fault Tolerance) as well as PBFT (Practical Byzantine Fault Tolerance) [3, 10]. Raft uses election mechanism to choose "Leader" among Orderer nodes.

This leader will interact with peers in the network for transaction replication and block creation [10].

- **High TPR.** The most prominent advantage of Hyperledger Fabric over Sawtooth is transaction processing rate (TPR), with TPR > 2000/ sec [10], compared to Swatooth's approx TPS (transaction per second) of over 1000 [3,10]. Indeed, the high TPR of Hyperledger Fabric is it's leading edge, which ensures high transaction throughput, scalability and making it more suitable for enterprises to adopt this platform [10,44].

- **Private Channel Creation**. Since SecureERP is deployed in a single-controlled organization that wants complete control over its network to ensure confidentiality of data. During analyzing the use-cases, we foresee the integration of SecureERP with other allied organizations in future. In such a scenario, SecureERP will require different private channels for communicating and exchanging data with allied organizations while maintaining data confidentiality. Hence, Hyperledger Fabric the only private Blockchain, which offers private channel creations in order to maintain secrecy, integrity, confidentiality in the network. This feature (Creation of private channel) is not typically offered by other Blockchain platform (like Sawtooth, Orion, etc.) [3,10,43]. Indeed, this is one of the primary reason, we preferred Hyperledger Fabric for SecureERP like enterprises, where data confidentiality is preferred over transparency.

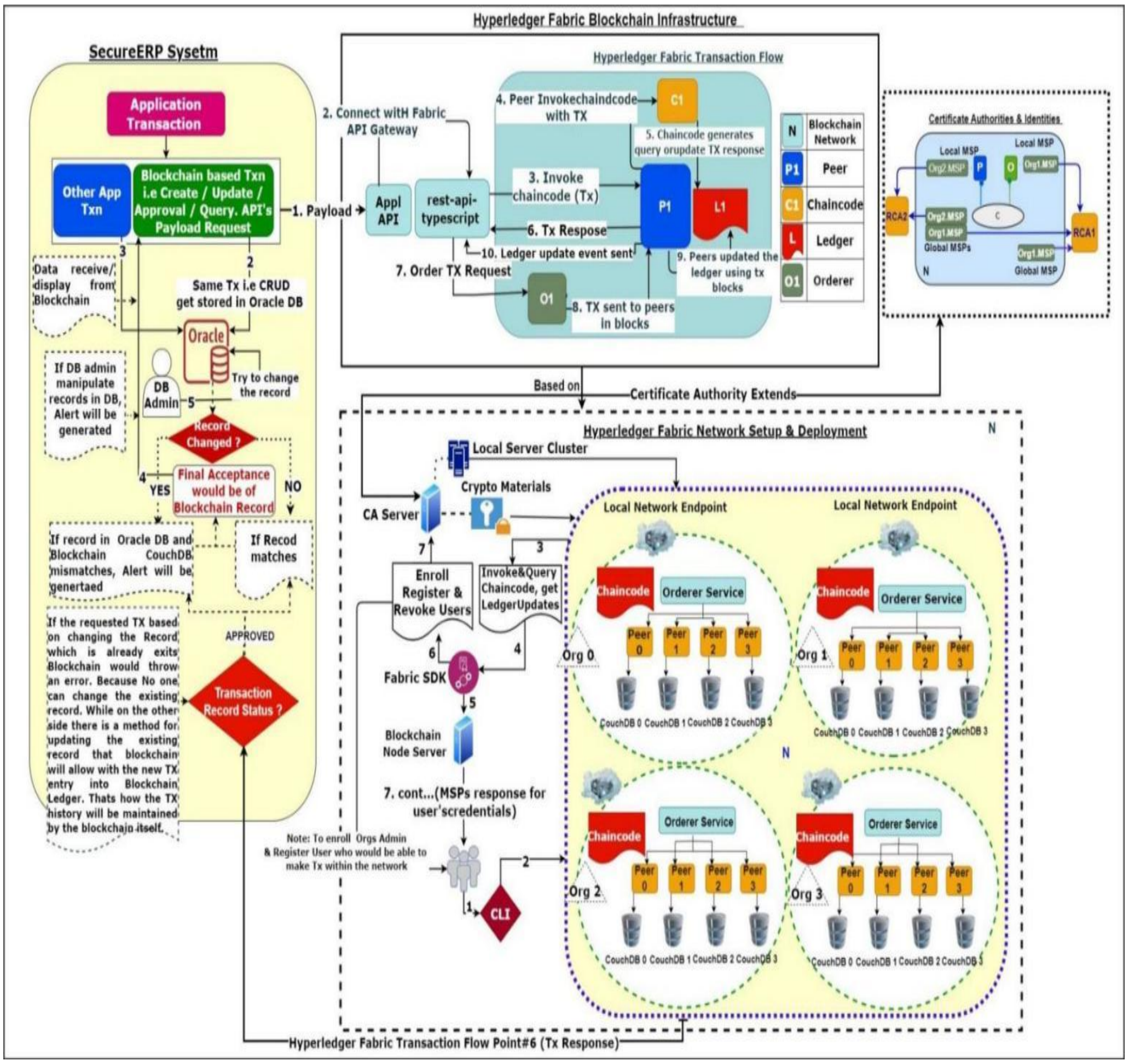
3.3 Proposed Architecture

Since we have discussed the reasons for adopting Hyperledger Fabric and analyzed the requirements of organizations—such as a daily transactional volume between **10,000-15,000,** and prioritizing data confidentiality over transparency. We now proposed a model and framework that fulfill these organizational requirements. The proposed model is depicted at **Figure-3.2**.

SecureERP is confined to a centralized Oracle database system. All the transactions such as, Create, Read, Update, Delete (CRUD operations) are processed within the Oracle database. In order to align with the transactional volume of SecureERP, we believe that Hyperledger Fabric will handle this workload efficiently. Moreover, two scenarios are possible to integrate SecureERP with the Hyperledger Fabric blockchain platform.

These scenarios are discussed in next sections. Enterprises must analyse their use-case and can adopt most suitable scenario for implementation. However, our proposed model will remain same for both the scenarios.

Proposed Architecture - Integration of SecureERP with Hyperledger Fabric Blockchain



Hyperledger Fabric Blockchain Infrastructure
SecureERP Sysetm
Hyperledger Fabric Transaction Flow
Application Transaction
Other App Txn
Blockchain based Txn i.e Create / Update / Approval / Query. API's Payload Request
1. Payload
2. Connect witH Fabric API Gateway
Appl API
rest-api-typescript
3. Invoke chaincode (Tx)
4. Peer Invokechaincode with TX
5. Chaincode generates query orupdate TX response
6. Tx Respose
7. Order TX Request
8. TX sent to peers in blocks
9. Peers updated the ledger using tx blocks
10. Ledger update event sent
C1
P1
L1
O1
N Blockchain Network
P1 Peer
C1 Chaincode
L Ledger
O1 Orderer
Certificate Authorities & Identities
Local MSP
Org2.MSP
Org1.MSP
Global MSPs
Global MSP
RCA2
RCA1
Data receive/ display from Blockchain
Same Tx i.e CRUD get stored in Oracle DB
Oracle
If DB admin manipulate records in DB, Alert will be generated
DB Admin
Try to change the record
Record Changed ?
YES
NO
Final Acceptance would be of Blockchain Record
If record in Oracle DB and Blockchain CouchDB mismatches, Alert will be genertaed
If Recod matches
If the requested TX based on changing the Record which is already exits Blockchain would throw an error. Because No one can change the existing record. While on the other side there is a method for updating the existing record that blockchain will allow with the new TX entry into Blockchain Ledger. Thats how the TX history will be maintained by the blockchain itself.
APPROVED
Transaction Record Status ?
Based on
Certificate Authority Extends
Hyperledger Fabric Network Setup & Deployment
Local Server Cluster
Crypto Materials
CA Server
Enroll Register & Revoke Users
Invoke&Query Chaincode, get LedgerUpdates
Fabric SDK
Blockchain Node Server
7. cont...(MSPs response for user'scredentials)
Note: To enroll Orgs Admin & Register User who would be able to make Tx within the network
CLI
Local Network Endpoint
Chaincode
Orderer Service
Peer 0
Peer 1
Peer 2
Peer 3
CouchDB 0 CouchDB 1 CouchDB 2 CouchDB 3
Org 0
Org 1
Org 2
Org 3
Hyperledger Fabric Transaction Flow Point#6 (Tx Response)

Figure 3. 2: Proposed Architecture for Production Environment

Figure 3.2, conceptualizes integration of SecureERP and Hyperledger Fabric using Scenario-2, The only participating organization is SecureERP itself.

3.4 Integration’s Scenarios

We perceived following two integration scenarios between SecureERP system and Hyperledger Fabric Blockchain, for both the scenarios, our proposed model (Figure-3.2) and framework will remain same.

3.4.1 Scenario -1 (Future Use-Case)

Presently, SecureERP lacks support for distributed transactions and Decentralized Application (DApps). As stated earlier that, SecureERP serves as a central repository for allied organizations within it network, the data are shared using traditional approaches such as Service-Oriented-Architecture (SOA) services, email etc with these organizations. Thus, the integration of SecureERP with these allied organizations using Hyperledger Fabric cannot be overlooked.

In Scenario-1, all participating or authorized organizations may adopt our proposed model, as illustrated in Figure 3.2. It is important to highlight that these participating organizations will be physical entities that have reached a consensus with SecureERP to interact over the blockchain or Hyperledger Fabric. Specifically, these organizations will perform transactions that can be securely verified by all these participating organization based on the consensus policy. However, in such a situation, the participating organizations have to devise a method or develop User Interface (UI) that allows them to make transactions over Blockchain Network. Surely, adopting this approach will require alteration at SecureERP level such as, chaincode development, distributed ledger, APIs, changes in application etc. We have discussed these challenges in details in the implementation section of this paper. Therefore, it is reiterated that analyzing use-case, business requirements, and cost involved is critical, before shifting existing ERP system into blockchain-based ERP. This graphical representation of scenario-1 is shown in Figure-3.3.

In the diagram (Figure-3.3), the Organizations such as SecureERP, Org A, Org B and Org C would be physical organizations participating in the Hyperledger Fabric Network. In the context of problem statement, the scenario-1 could be the future use-case, where all these organizations necessitate to interact with each other using blockchain or Hyperledger Fabric for transaction immutability and integrity purpose, there by to avoid any fraudulent activities in the transactional data.

Presently, SecureERP does not interact with these allied organizations using blockchain, instead, it uses traditional method for sharing data.

Integration Scenario -1

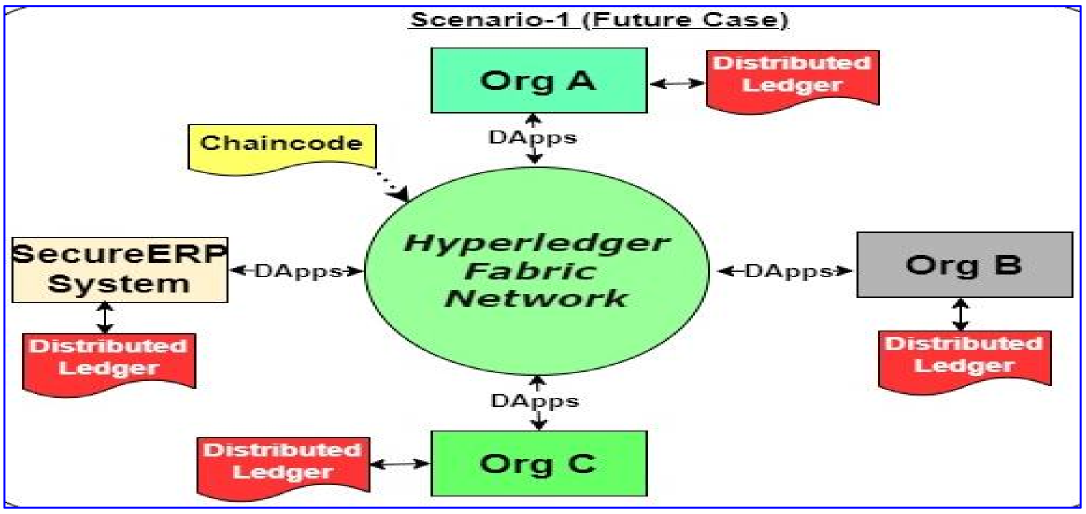


Figure 3.3: Scenario -1

3.4.2 Scenario -2 (Present Use-Case)

The second scenario outlines the current working procedure or environment of SecureERP system. It means, SecureERP is the only physical participant or organization interacting with Hyperledger Fabric blockchain. Here, the aim of integrating SecureERP and Hyperledger Fabric is to have an immutable, auditable ledger that can be referenced for audit purposes. Moreover, we have already discussed that data in SecureERP is susceptible to unwanted changes by insiders or database administrators.   So the proposed model should restrict these administrators from an illegitimate transactions or manipulation in the Oracle database.

Our model is adaptable, in future, if SecureERP system is required to implement or adopt Scenario-1 due to organizational needs, we assured that due to modular and flexible nature of of Hyperledger Fabric, SecureERP can easily switch to Scenario-1 at any point of time with minimal effort. However, we have conducted our POC specifically for Scenario-2. The graphical representation Scenario-2 is depicted at Figure-3.4

<u>Scenario -2 (Present Operational Environment)</u>

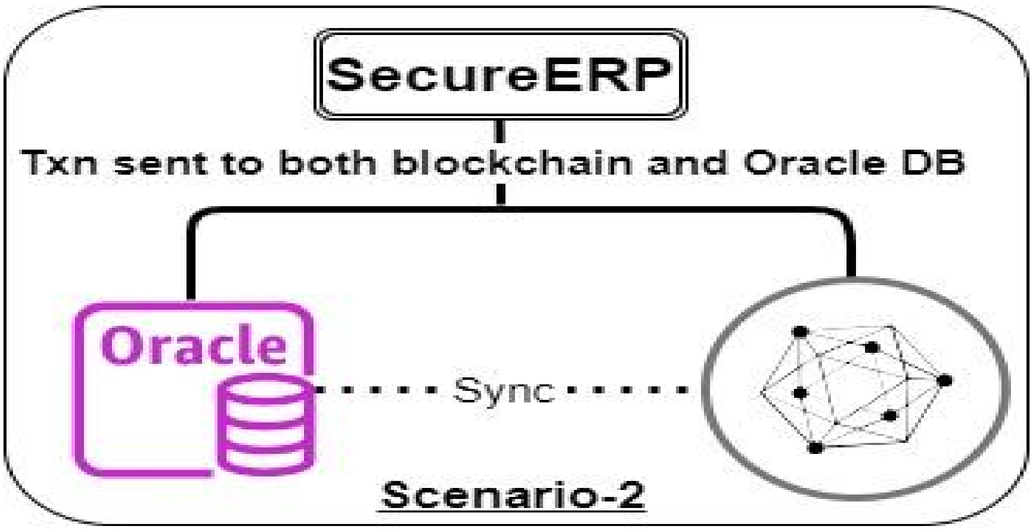


Figure - 3.4 : Scenario-2

3.5 Transaction Synchronization - Oracle Database and Hyperledger Fabric

Since, our use-case is based on Scenario-2, therefore, we conducted POC to validate our proposed model using scenario-2. In our proposed model, transaction is recorded in the Oracle database (SecureERP), and in the CouchDB (Hyperledger Fabric) using REST API and Fabric Gateway. The transaction flow is elaborated in Figure-3.2 in details. Transactions recorded in the Hyperledger Fabric are immutable and temper-proof, and can be used for audit purposes. However, the same transactions in the Oracle Database (SecureERP) can be manipulated. To address this vulnerability, and to restrict database administrators or insiders from an illegitimate activities at Oracle database, our model synchronizes and validating transaction records between the two systems. For example, if the database administrator (insider) or unauthorized person manipulates an approved transaction in the Oracle database, our model will compare and validate the manipulated transaction with blockchain record, and it will generate alerts if there is a mismatch between transaction in Hyperledger Fabric and Oracle database. We believe that the synchronization of both the systems (i.e. SecureERP's Database and Blockchain Data) will enforce system to maintain integrity and consistency of data. Though, transactions logs are kept in both the systems, however, records stored in the hyperledger fabric blockchain would be considered for final acceptance or audit purposes.

3.6 Design Analysis

In this section, we critically analyze the proposed model in terms of performance, the roles of various components, and their configuration within the Hyperledger Fabric Network. Our aim is to implement scenario-2, where SecureERP is the only physical organization interacting with the Hyperledger Fabric Network. Hyperledger Fabric is a decentralized, Permissioned blockchain platform, which requires the participation of atleast two organizations to operate in the blockchain network. The reason behind this is to ensure decentralizing nature of hyperledger fabric.

In our proposed model (Figure-3.2), we have configured and recommended four logical organizations specifically for SecureERP. These four logical organizations will simulate a multi-organizational setup, thereby maintaining the decentralized nature of Hyperledger Fabric.

3.6.1 Hyperledger Fabric Nodes Requirement

SecureERP is a robust application, designed to support a high daily transactional volume (10,000 to 15,000 transactions), and providing services to a large user base. The scalability and resiliency in Hyperledger Fabric network, could be achieved using clusters of Peer nodes, Orderer and having multiple participating organization in the blockchain network. We have suggested number of nodes (Peer, Orderer) required to integrate SecureERP and Hyperledger Fabric in production environment are shown in Table-3.1.

Overview of Nodes - SecureERP Production Environment

Node Type	No of Nodes	Purpose
Peer	16	• Transactions endorsement and commitment. • We have four logical organizations operating at Blockchain Network, having four (4) peer at each.
Orderer	5	• Transactions' sequencing, creating and replicating blocks • Raft-based consensus
CA	4*	• Identity management, each Organization has it own CA Server. • * if the four logical organization are deployed at different geographical locations/ data centers

Table-3.1 : Required number of Nodes

3.6.2 Number of Participating Organizations

In Section 3.4, we presented two integration scenarios. Our use-case is based on scenario-2, where a single controlled organization (SecureERP) is integrated with the Hyperledger Fabric. Since, Hyperledger Fabric is a decentralized, Permissioned blockchain platform. In order to maintain the decentralized nature of Hyperledger Fabric,we have deployed four organizations within the Hyperledger Fabric framework. It's worth mentioning that these organizations are logical, since we are implementing scenario-2. We further advise to deploy or configure these four logical organizations at different geographic locations or data centers for high availability. The recommendation of four organizations with four peer nodes in each (Table-3.1), is considering to align with the transactional volume of SecureERP. This deployment model will ensure high availability of the blockchain network and mitigates the risk of single failure. So, whenever a transaction is committed in SecureERP, same transaction will be replicated among these four organizations (distributed ledger), i.e. Four additional copies of the same transactions will be recorded at Blockchain level. This will increase the HW cost as well as complexity. We will discuss this in details later in the challenges section.

It is important to note that the structure and framework of our proposed model will remain same irrespective of the type of participating organizations, whether they are logical or physical entities. If Organization where SecureERP is deployed, decides to implement Scenario-1, as discussed in section 3.4.1, the four logical organizations do not need to be deployed. Instead, multiple physical entities, with numbers based on business requirements, will participate in the blockchain network and can make transactions with the consent of each participating organization, including SecureERP. Our model is flexible, enabling SecureERP to adapt to the specific needs and structures of different business scenarios.

3.6.3 Orderer Service Configuration

As discussed in [10,43], "Cluster of Orderer nodes create or provide Ordering service". The responsibility of Orderer Service or nodes is to ensure sequencing and ordering of transactions in the block. Orderer node doesn't have chaincode to execute client transactions, it receives transactions that are endorsed by endorsing peer, and order these endorsed transactions into the block [49]. The blocks are sent to all committing peers for validation and replicating in the distributed ledger [49]. Orderer nodes ensure the integrity and consistency of transactions in Hyperledger Fabric by maintaining correct order or sequence of transactions with all peers [49]. It is recommended to deploy Orderer nodes in a cluster, specially in the production environment for high availability, load balancing and to avoid single failure risk.

Deploying Orderer in a cluster, will ensure fault tolerance (FT) and high availability (HA) of our Hyperledger Fabric network. However, considering number of Orderer nodes in the cluster is critical. Increasing the number of Orderer nodes in the cluster will drastically decrease the performance (high latency and low TPR). The reason behind this is simple, when we increase the number of Orderer nodes, this means, we are increasing the participants in the consensus mechanism. So, a balance to be maintained between FT , latency and TPR. We use the following formula to calculate fault tolerance in the cluster:-

$$\mathbf{FT = (N-1)/2}$$

" N is the total number of Orderer nodes in the cluster".

In our proposed model, we suggested five (5) Orderer nodes for SecureERP system, giving FT=2 , This shows that, if 2 Orderer nodes get down, then the remaining three Orderer nodes will still be able to operate the network [10,42,43]. Moreover, it's advised to use an odd number of Orderer nodes in the cluster, this will prevent scenarios where the network splits into equal halves during partitions.

Unlike other blockchain platform, Hyperledger Fabric comes with a modified transaction flow. The Orderer node in the Hyperledger Fabric doesn't have chaincode to execute, instead chaincode is installed at endorsing peers only. Thus, endorsing peer will execute transaction, but before writing into ledger, transactions are sent to Orderer Service for block creation.

Separating Orderer Service from the peers has increased the performance and scalability of Hyperledger Fabric [10,43, 44].

3.6.4 Raft-Based Consensus Mechanism

Hyperledger Fabric uses a multi-layered process to ensure that a valid transaction is added to the block [51]. This includes several steps such such check permissions, endorsements, data synchronization among participants, transaction order, and the accuracy of changes. The consensus in Fabric is divided into three phases as follow [51]. The graphical representation is shown at Figure-3.5. The three steps include:-

- **Endorsement**. Steps 1–2 in Figure-3.5, describe the endorsing process in Hyperledger Fabric. In our model, we have four logical organizations, where each organization has two endorsing peer and two committing peers. For readability, in the Figure-3.5, we have shown a single endorsing peer in each organization [51].
- **Ordering .** Steps 3–4 in Figure-3.5, describe the actions or tasks perform by Orderer Service in our model. The aim of this step is to ensure, sequence, and validity of transactions in the block [51].
- **Validation and Commitment.** Step 4-5 also describe the response of transaction and delivered to the client [51].

The Orderer service uses Raft consensus mechanism. An Orderer node can be of type “Leader, Candidate or Follower”. Unlike PoET, where each validtor node gets random amount of time to create block in the blockchain, Raft uses election process to elect leader among Orderer nodes in the cluster. The leader will be responsible to create blocks and communicate with other peer nodes for replication of blocks in the blockchain. The election process and selection of leader in the cluster can be excellently simulated through“**thesecretlivesoftdata.com/raft/**”. A leader node uses Gossip protocol for replication of logs and messaging among other nodes in the network [10].

Raft-Based Consensus Mechanism

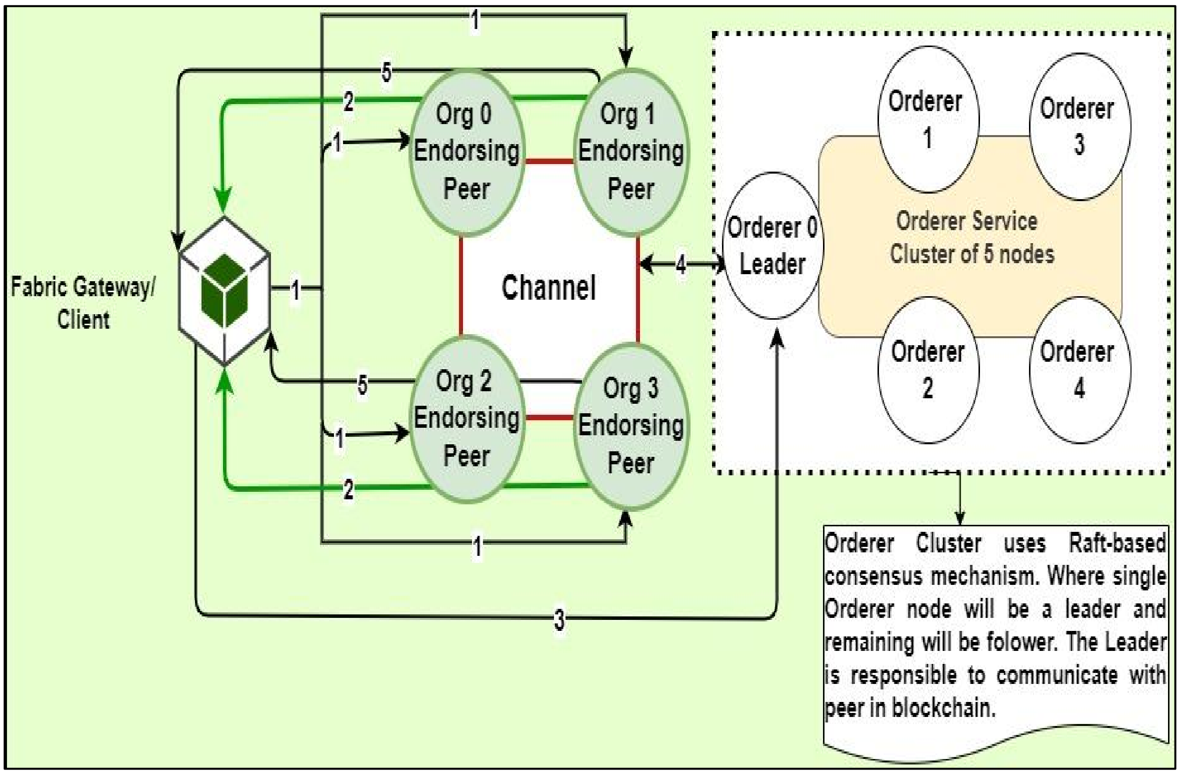
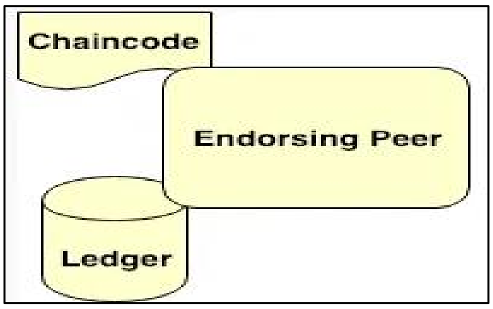
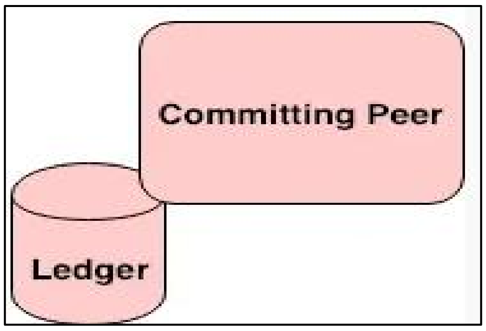

Fabric Gateway/ Client
Org 0 Endorsing Peer
Org 1 Endorsing Peer
Org 2 Endorsing Peer
Org 3 Endorsing Peer
Channel
1
2
3
4
5
Orderer 0 Leader
Orderer 1
Orderer 2
Orderer 3
Orderer 4
Orderer Service Cluster of 5 nodes
Orderer Cluster uses Raft-based consensus mechanism. Where single Orderer node will be a leader and remaining will be folower. The Leader is responsible to communicate with peer in blockchain.

Figure : 3.5 (Source: https://medium.com/coinmonks/)

Figure 3.5, describes the consensus and endorsing mechanism in our proposed model

3.6.5 Peer Configuration and Block Size





We proposed four logical participating organizations with four peer nodes in each, shown in figure-3.2. Broadly, peer can be of type “Endorser” or “Committer”. Endorser maintains copy of ledger and chaincode installed on it, while committer does not have any chaincode, and only maintain a ledger. To ensure fault tolerance and high availability, we suggested two endorsers and two committer in each organization. It is necessary to mention that, client has to interact with endorser via Fabric Gateway [10].

Block Size: By default, Hyperledger Fabric use 512 KB, however, maximum size of block is allowed up to 99 MB [10]. (Diagrams Source [51])

3.6.6 Performance Considerations

We already discussed in the introduction section that Hyperledger Fabric is highly modular blockchain platform, which allows plug-play functionalities. So, the overall performance depends on the configuration factors, like number of channels, number of participating organizations and Orderer nodes, chaincode complexity , transaction flow and block-size [10,44]. While there's no one-size-fits-all recommendation, we would suggest to relate official documentation on Performance, Hardware consideration before deploying Hyperledger Fabric at production level. In this paper, we mentioned information about Peer node at Table-2, for remaining nodes like Orderer, CA, MSP etc, documents available at "https://hyperledger-fabric.readthedocs.io/en/release-2.5/performance.html" must be referred [44].

Peer Node - System Specification

CPU	Multi Core Processors, minimum 8 cores, recommended 16 cores or more
RAM	Minimum 16 GB, recommended more
Storage	SSD (Solid State Drive) storage is preferred for faster access to ledger data. Allocate enough storage based on anticipated blockchain data growth
Operating System	Linux distributions (e.g., Ubuntu, CentOS), macOS, or Windows
Network	Ensure a high-bandwidth and low-latency network connection between Hyperledger Fabric nodes for smooth communication

Table-3.2

Source: www.hyperledger.com

3.7 Summary

In Chapter 3, we present the design of our proposed model, that integrates SecureERP system with Hyperledger Fabric. We explore two possible integration scenarios for SecureERP system, keeping its present and future use-cases. Regardless of the scenario, our proposed model will remain same. We also analyzed the design's impact on participating organizations and the number of nodes involved. To ensure optimal performance in the production environment, we recommend to follow the detailed documentation provided by Hyperledger Fabric regarding hardware specifications and performance considerations.

CHAPTER 4

IMPLEMENTATION OF

PROPOSED MODEL

4.1 Overview

The Chapter discusses the imp[implementation of our proposed architecture as depicted at Figure-3.2 . The main objective of this section, to develop a prototype that validates and tests our proposed model as conceptualized in the design section (Chapter 3). We adopted a comprehensive strategy that involves creating an environment to test the developed prototype. The environment and the implementation details are provided in the following sections.

4.2 Environment

The environment are created on a machine equipped with an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz (up to 2.90 GHz) and 16 GB of RAM running Windows 10 Pro. To validate the end-to-end functionality of the model, three virtual servers are created, which are discussed briefly in below sections.

4.2.1 Infrastructure Setup (Virtual Machines)

We created an environment with necessary infrastructure, which comprises deploying of following three Virtual machines (VM) servers:-

- **Application Server VM**. This is windows-based VM (Virtual box) and hosting Java-based custom application that acts as an interface for SecureERP Database (in our case) and Blockchain network.
- **Oracle Database VM**. This is another windows-based VM, hosting Oracle Database 21c and acts as a back end server for Java based application (SecureERP application). In our model, transactions are stored in Oracle database and Hyperledger Fabric's CouchDB as well. We then synchronize records in both the systems, our model generates alerts, if mismatch found.
- **Hyperledger Fabric Network**. This VM is Ubuntu based, where Hyperledger Fabric network is deployed. We deployed the latest version of Hyperledger Fabric

(V2.5) with CouchDB as State Database, and Raft protocol for consensus mechanism is used.

4.2.2 Java-based Application

In order to validate the functionalities and test our model in true letter and spirit, front-end application development was essential. Without this application, testing and validating the model through use cases and scenarios wouldn't be possible. Therefore, we designed and developed a Java-based application that demonstrates the core functionalities of a SecureERP system. This application is used as a front-end for both SecureERP environment (Oracle database 21c) as well as for Hyperledger Fabric blockchain network. The key processes in this applications are "Item Creation, Purchase Order, Issue Order, Transfer Order and Stock Position Dashboard etc". The Home Page (after login) of this Java-based application is shown at Figure-4.1. This Java-based application is hosted at window-based VM as discussed in previous section.

<u>User Interface (UI) - Java based Application</u>

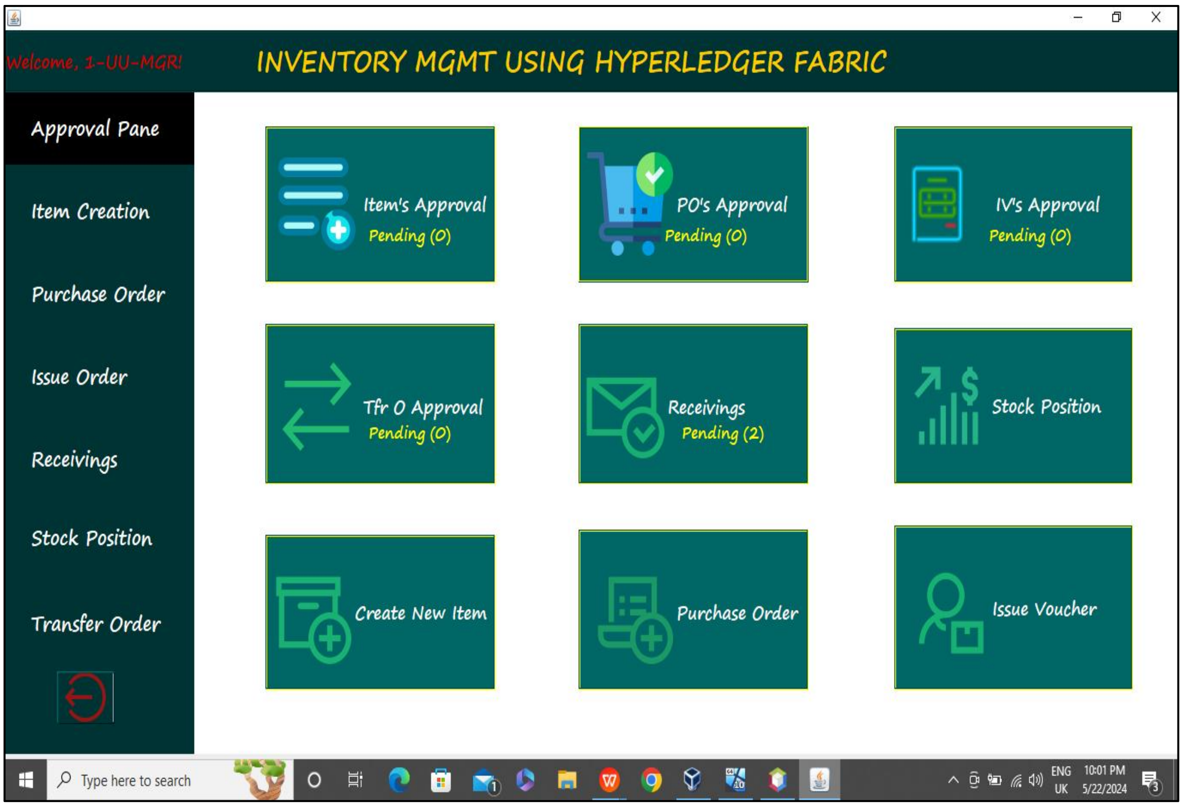


Figure-4.1

4.2.3 Hyperledger Fabric Deployment

We deploy Hyperledger Fabric with Sample-repository. The samples repository provides a Docker-Compose based test network with two Organizations (**Org 0** and **Org 1**) having single peer in each organization, and an ordering service node [10]. To use the Fabric samples, we need to download the Fabric Docker images and the Fabric CLI tools. But before cloning or deploying Hyperledger Fabric, we have to to ensure that all the prerequisites are installed. The prerequisites for each version can be found at "https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html". After this, we can follow the instructions at "https://hyperledger-fabric.readthedocs.io/en/latest/install.html" to install the Fabric samples repository, Binaries, and Docker Images. In addition to downloading the Fabric images and binaries, the Fabric samples will also be cloned to local machine [10]. List of prerequisites along with commands used for installation purpose are shown at Table 4.1.

Hyperledger Fabric is the backbone of our proposed solution, responsible to maintain an immutable transactions logs and enusre ERP transaction's integrity. We deployed latest version V2.5 of Hyperledger Fabric (using command, ***curl -sSL https://bit.ly/2ysbOFE | bash -s***). We configured two organizations having one peer in each organization, one Orderer node, two CouchDB instances with each peer in each organization, and Fabric CA (Certificate Authorities) for issuing certificates. Docker containers / images of our deployed test-network are shown at Figure-4.2. The Hyperledger Fabric blockchain network is deployed on Ubuntu-based virtual server.

We have already discussed in the design section that Hyperledger Fabric is a decentralized, Permissioned Blockchain platform that requires atleast two participating organizations (Logical or physical depends on use-cases) in the newtrok to ensure the decentralized nature of Fabric network. In our test-network, we have two organizations, but we have recommended four organizations for SecureERP at production environment.

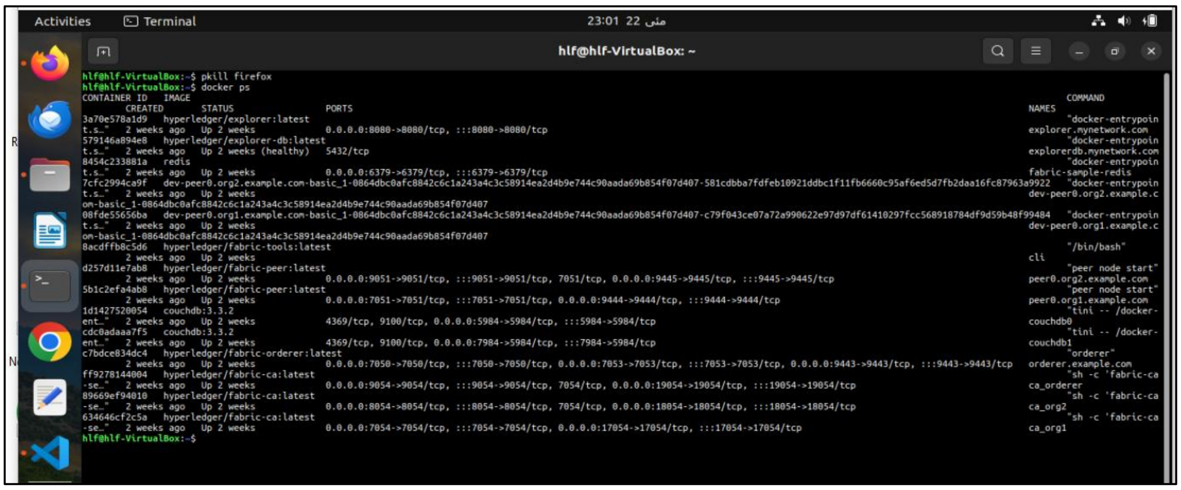
Prerequisites - Hyperledger Fabric Deployment

S.No	Module	Command
1.	***cURL -*** Stands for Client URL. It is a command-line tool and library, used for transferring data with URLs. It supports various protocols, including HTTP, HTTPS, FTP, SFTP, SCP, and more.	Run below commands to install cURL and check its version ● Sudo apt-get install curl ● curl --version
2.	**NodeJS -** We have installed NodeJS for writing our Chaincode in TypeScript/ NodeJS/ Java Script	● curl-sL https://deb.nodesource.com/setup_10.x | sudo -E bash – ● sudo apt-get update ● sudo apt-get install nodejs ● node --version
3.	Git Installation	● sudo apt-get install git ● git --version
4.	Python Installation	● sudo apt-get install python3 ● python --version
5.	Lib Tools Installation	● sudo apt-get install libltdl-dev
6.	Install Docker CE (Community Edition)	First download then install it using below commands ● Wget https://download.docker.com/linux/ubuntu/dists/xenial/pool/ stable/amd64/docker-ce_18.06.3~ce~3-0~ubuntu_amd64.deb

		• sudo dpkg -i docker-ce_18.06.3~ce~3-0~ubuntu_amd64.deb • docker --version
7.	Install Docker Compose	• sudo apt-get install python-pip • pip --version • sudo pip install docker-compose • docker-compose version
8.	Go Language	
9.	Hyperledger Installation (Sample Network repository will be cloned to local machine)	● curl -sSL https://bit.ly/2ysbOFE | bash -s

Table-4.1

<u>List of Running Docker Images/ Containers (docker ps -a)</u>



```
hlf@hlf-VirtualBox:~$ pkill firefox
hlf@hlf-VirtualBox:~$ docker ps
CONTAINER ID   IMAGE                                                                                                                                                                        COMMAND
       CREATED       STATUS                 PORTS                                                                                                                                      NAMES
3a70e578a1d9   hyperledger/explorer:latest                                                                                                                                                  "docker-entrypoin
t.s…"   2 weeks ago   Up 2 weeks             0.0.0.0:8080->8080/tcp, :::8080->8080/tcp                                                                                                  explorer.mynetwork.com
579146a894e8   hyperledger/explorer-db:latest                                                                                                                                               "docker-entrypoin
t.s…"   2 weeks ago   Up 2 weeks (healthy)   5432/tcp                                                                                                                                   explorerdb.mynetwork.com
8454c233881a   redis                                                                                                                                                                        "docker-entrypoin
t.s…"   2 weeks ago   Up 2 weeks             0.0.0.0:6379->6379/tcp, :::6379->6379/tcp                                                                                                  fabric-sample-redis
7cfc2994ca9f   dev-peer0.org2.example.com-basic_1-0864dbc0afc8842c6c1a243a4c3c58914ea2d4b9e744c90aada69b854f07d407-581cdbba7fdfeb10921ddbc1f11fb6660c95af6ed5d7fb2daa16fc87963a9922   "docker-entrypoin
t.s…"   2 weeks ago   Up 2 weeks                                                                                                                                                        dev-peer0.org2.example.c
om-basic_1-0864dbc0afc8842c6c1a243a4c3c58914ea2d4b9e744c90aada69b854f07d407
08fde55656ba   dev-peer0.org1.example.com-basic_1-0864dbc0afc8842c6c1a243a4c3c58914ea2d4b9e744c90aada69b854f07d407-c79f043ce07a72a990622e97d97df61410297fcc568918784df9d59b48f99484   "docker-entrypoin
t.s…"   2 weeks ago   Up 2 weeks                                                                                                                                                        dev-peer0.org1.example.c
om-basic_1-0864dbc0afc8842c6c1a243a4c3c58914ea2d4b9e744c90aada69b854f07d407
8acdffb8c5d6   hyperledger/fabric-tools:latest                                                                                                                                              "/bin/bash"
       2 weeks ago   Up 2 weeks                                                                                                                                                         cli
d257d11e7ab8   hyperledger/fabric-peer:latest                                                                                                                                               "peer node start"
       2 weeks ago   Up 2 weeks             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
5b1c2efa4ab8   hyperledger/fabric-peer:latest                                                                                                                                               "peer node start"
       2 weeks ago   Up 2 weeks             0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp                                                       peer0.org1.example.com
1d1427520054   couchdb:3.3.2                                                                                                                                                                "tini -- /docker-
ent…"   2 weeks ago   Up 2 weeks             4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp, :::5984->5984/tcp                                                                              couchdb0
cdc0adaaa7f5   couchdb:3.3.2                                                                                                                                                                "tini -- /docker-
ent…"   2 weeks ago   Up 2 weeks             4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp, :::7984->5984/tcp                                                                              couchdb1
c7bdce834dc4   hyperledger/fabric-orderer:latest                                                                                                                                            "orderer"
       2 weeks ago   Up 2 weeks             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
ff9278144004   hyperledger/fabric-ca:latest                                                                                                                                                 "sh -c 'fabric-ca
-se…"   2 weeks ago   Up 2 weeks             0.0.0.0:9054->9054/tcp, :::9054->9054/tcp, 7054/tcp, 0.0.0.0:19054->19054/tcp, :::19054->19054/tcp                                         ca_orderer
89669ef94010   hyperledger/fabric-ca:latest                                                                                                                                                 "sh -c 'fabric-ca
-se…"   2 weeks ago   Up 2 weeks             0.0.0.0:8054->8054/tcp, :::8054->8054/tcp, 7054/tcp, 0.0.0.0:18054->18054/tcp, :::18054->18054/tcp                                         ca_org2
634646cf2c5a   hyperledger/fabric-ca:latest                                                                                                                                                 "sh -c 'fabric-ca
-se…"   2 weeks ago   Up 2 weeks             0.0.0.0:7054->7054/tcp, :::7054->7054/tcp, 0.0.0.0:17054->17054/tcp, :::17054->17054/tcp                                                   ca_org1
hlf@hlf-VirtualBox:~$
```

Figure-4.2

Figure-4.2, shows Hyperledger Fabric is successfully deployed

4.2.4 REST API

REST (Representational State Transfer) is an architectural style that is used to transfer data over the web. RESTful APIs use common HTTP methods such as, GET, POST, PUT, DELETE for data handling. In our model, we used RESTful APIs to facilitate data exchange between the SecureERP system and Hyperledger Fabric's CouchDB database. The data is exchanged in JSON (JavaScript Object Notation) format. These HTTP methods enable us to perform various CRUD (Create, Read, Update, Delete) operations on the blockchain data. We will see the desired role and action of each HTTP method in the POC section later in this paper.

In this paper, we only explained GET HTTP method (with Code Snippet) to understand the execution cycle of REST API particularly in our model. This will make it clear, how our application uses REST API to interact with Hyperledger Fabric Blockchain network.

- **GET Method**. We have used GET HTTP method to retrieve data from blockchain (Hyperledger Fabric - CouchDB instance). The retrieved records are then displayed on Front-end application for end-user visualization. The steps below represent the execution of GET HTTP method in our proposed model.
  - **Step-1 : Initiating GET Request**. The Java front-end application initiates the data retrieval process by invoking a function or method that sends a GET request through the RESTful API. A snippet from the Front-end application is shown in Table-4.2.
  - **Step-2 : REST API**. In this step, the received request is parsed by RESTful API in our Hyperledger Fabric Network (Ubuntu -based Machine). The API then interact with chaincode and called the required method written in the chaincode to retrieve data. In our case, "***ReadAllItems***" method is called to

get data from blockchain by invoking chaincode. A snippet from REST API is shown in Table-4.3

- **Step-3 : Chaincode Method**. We have written our chaincode in TypeScript. The REST API using Fabric Gateway invoke the chaincode, the chaincode method retrieves the specified data from CouchDB based on the requested parameters. For simplicity and understandability purpose, we have shown "**ReadAllItem**s" method in our chaincode. This method returns records of all transactions or record committed to blockchain network. A snippet is shown in Table-4.4

- **Step-4 : Display Result on Front-end Application.** The Chaincode method will returns all records based on parameter to front-end application for user visualization. Screenshot is shown at Figure-4.3

GET Method URI : Front-end Application

Step-1

```
/*  This method  is fetching record from Blockchain (CouchDB) and displays on the
Front-end application */
Private void setRecordtoTable() {
  String[] columnNames = {"Item No", "Item Name", "Selling Price", "Buying Price",
  CreatedBy","Updated By", "Created Date", "Updated Date", "Status","CheckBox" };
                      //--------Exiting Code------------removed for clarity
try {           /*  URI Calling REST API    */
URL apiUrl = new URL(BASE_API_URL + "/api/items/");
HttpURLConnection getConnection = (HttpURLConnection) apiUrl.openConnection()
 // GET HTTP method called here
getConnection.setRequestMethod("GET");
getConnection.setRequestProperty("Content-Type", "application/json");
                                    // --------Remaining Code------
```

Table -4.2

REST API CODE - EXAMPLE

Step-2 :

This piece of code is taken from our REST API, in this code a Single method "GetAllItems" is demonstrated. This method is written in chaincode, we are invoking this method using REST API to retrieve data from blockchain network.

-- REST API code--

```
import express, { Request, Response } from 'express';
import { body, validationResult } from 'express-validator';
import { Contract } from 'fabric-network';
import { getReasonPhrase, StatusCodes } from 'http-status-codes';
import { Queue } from 'bullmq';
import { AssetNotFoundError } from './errors';
import { evatuateTransaction } from './fabric';
import { addSubmitTransactionJob } from './jobs';
import { logger } from './logger';
const { ACCEPTED, BAD_REQUEST, INTERNAL_SERVER_ERROR,
NOT_FOUND, OK } = StatusCodes;
export const itemsRouter = express.Router();
itemsRouter.get('/', async (req: Request, res: Response) =>
{logger.debug('Get all items request received');
try {const mspId = req.user as string;
        const contract = req.app.locals[mspId]?.assetContract as Contract;
// GetAllItems is a method in chaincode that retrieves saved record in --Hyperledger
Fabric.
        const data = await evatuateTransaction(contract, 'GetAllItems');
```

```
          let items = [];
          if (data.length > 0) {
          items = JSON.parse(data.toString());
          }
          return res.status(OK).json(items);
          } catch (err) {
          logger.error({ err }, 'Error processing get all items request');
          return res.status(INTERNAL_SERVER_ERROR).json({
          status: getReasonPhrase(INTERNAL_SERVER_ERROR),
timestamp: new Date().toISOString(),});
}});
```

Table - 4.3

Chaincode Snippet - GetAllItems Method

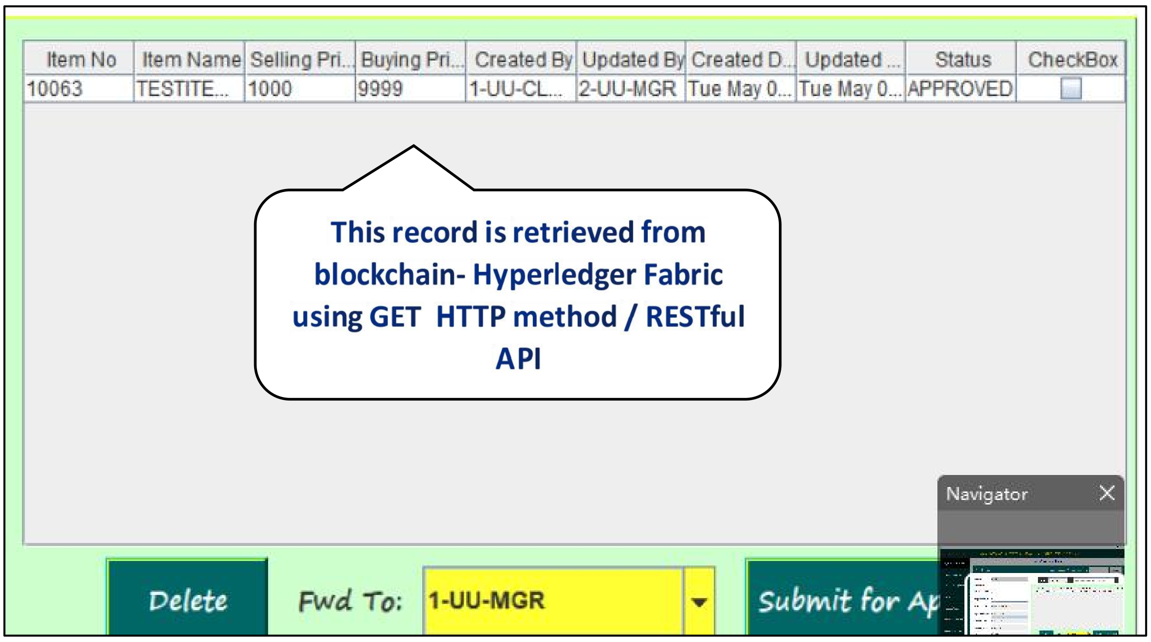
Step-3 ***// GetAllItems returns all items found in the world state.***

```
/*   SPDX-License-Identifier: Apache-2.0  */

// Deterministic JSON.stringify()

import {Context, Contract, Info, Returns, Transaction} from 'fabric-contract-api';

import stringify from 'json-stringify-deterministic';

import sortKeysRecursive from 'sort-keys-recursive';

@Info({title: 'ItemManagement', description: 'Smart contract for items management'})

export class ItemManagementContract extends Contract {

                    /*-------------------------- Exiting code-------------------*/
```

```
  @Transaction(false)

  @Returns('string')

  public async GetAllItems(ctx: Context): Promise<string> {

    const startKey = '0';

    const endKey = '999999';

    const allResults = [];

    const iterator = await ctx.stub.getStateByRange(startKey, endKey);

    let result = await iterator.next();

    while (!result.done) {

      const strValue = Buffer.from(result.value.value.toString()).toString('utf8');

      let record;

try {

        record = JSON.parse(strValue);

      } catch (err) {

        console.log(err);

        record = strValue;

      }

      allResults.push(record);

      result = await iterator.next();

    }

    return JSON.stringify(allResults);

  } }
```

Table - 4.4

Step-4 : Display Result on Front-end Application



Item No	Item Name	Selling Pri...	Buying Pri...	Created By	Updated By	Created D...	Updated ...	Status	CheckBox
10063	TESTITE...	1000	9999	1-UU-CL...	2-UU-MGR	Tue May 0...	Tue May 0...	APPROVED	☐

This record is retrieved from blockchain- Hyperledger Fabric using GET HTTP method / RESTful API
Navigator
Delete
Fwd To:
1-UU-MGR
Submit for Ap

Figure-4.3

- **POST Method**. We used **POST** method to insert new transaction into the Hyperledger Fabric (CouchDB). The POST method is referenced within the code associated with "**Save**" Button in our Front-end application. The Save button, saves records in the Oracle Database (Back-end database for SecureERP application). So, when a user clicks on the Save button, record will be saved to Oracle Database. Meanwhile, the POST method is also initiated, which will send the payload (Txn) into Hyperledger Fabric blockchain. In short, POST method is responsible for packaging the user-submitted record and transmitting it securely to the blockchain network.

The execution cycle of POST method is similar as we have seen for GET method in the previous section. However, the purpose and method in the chaincode will be different. i.e. GET retrieves data from the blockchain, whereas POST sends data to be stored on the blockchain. Moreover, a separate method in our Chaincode is written for POST method, which will handle the logic of storing the records in CouchDB (Hyperledger Fabric database).

- **PUT Method.** In the SecureERP (Front-end) application, when a transaction is updated using “Update Button”, the PUT HTTP method is used to send payload regarding the updated operation to the blockchain (Hyperledger Fabric). Each time an update occurs, a new transaction is created in the blockchain to capture the history of the update operation. This ensures that transactions history is maintained, tracked and immutable. We will see the details in the POC section.

- **DELETE Method.** Like an update, user can also delete records from SecureERP system. Records in the Oracle Database or SecureERP can be deleted, but the same record would exist in the blockchain for audit purposes. The HTTP or RESTful DELETE method is initiated , when users click on “Delete Button” in our front-end application. This step is demonstrated in the POC section.

4.2.5 Redis Server

Generally, Redis is used for performance and caching purpose [42]. Hyperledger Fabric supports the use of Redis server in its architecture. We used Redis to enhance the performance of our REST APIs. We use the following commands in Ubuntu terminal to start the Redis server in our Hyperledger Fabric Network:-

Starting Redis Server in Hyperledger Fabric

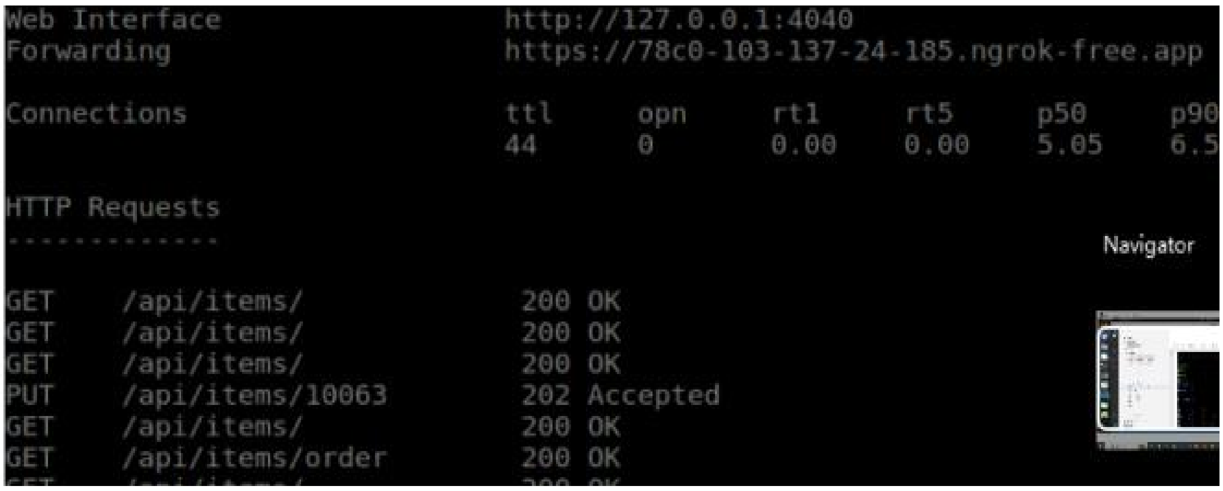
```
- - Start the Redis server:
$  export REDIS_PASSWORD=$(uuidgen)
$ npm run start:redis
If there is an issue of address binding while Starting the Redis then run the below
    command:-
$ sudo systemctl stop redis
Start the server:-
$ npm run start:dev
```

4.2.6 Ngrok

To facilitate seamless communication between our VM servers, we used Ngrok, which create secure path or tunnel/ URL to our blockchain network [47]. The Ngrok makes our Blockchain Server available over Internet without needing public IP or domain name [47]. we use the following command to run Ngrok in our environment, it will gives a URL as depicted at Figure-4.4

Command to run or start Ngrok

$ ngrok http 3000



```
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://78c0-103-137-24-185.ngrok-free.app

Connections                   ttl     opn     rt1     rt5     p50     p90
                              44      0       0.00    0.00    5.05    6.5

HTTP Requests
-------------

GET    /api/items/             200 OK
GET    /api/items/             200 OK
GET    /api/items/             200 OK
PUT    /api/items/10063        202 Accepted
GET    /api/items/             200 OK
GET    /api/items/order        200 OK
```

Figure-4.4

4.3 Summary

Implementation of complete proposed model is discussed in this chapter. We began with development of Java-based application, which is used to exhibits basic functionalities in the SecureERP application. After this, we discussed the infrastructure and environment, where we created three VM to test our model. Subsequently, we discussed RESTful APIs and Ngrok, which act as a bridge between SecureERP and Hyperledger Fabric blockchain network. In next chapter, we will conduct a Proof-of-Concept (POC) to validate our proposed model and its results within the implemented environment.

CHAPTER 5

POC

RESULTS AND ANALYSIS

5.1 Overview

The results section of our study is important component, which presents the findings of our proposed model. In this chapter, we present a Proof-of-Concept (POC) and the subsequent analysis of the results of our proposed model. The POC is necessary step in validating the practical feasibility, performance, and security of the proposed model. We have created a sample test-case that will validate end-to-end integration scenarios between the SecureERP and Hyperledger Fabric Blockchain. By the end of this chapter, we will have a clear understanding of how well the SecureERP integrates with Hyperledger Fabric to ensure the integrity of transactions.

To conduct POC, we developed Front-end application (Java-based application in NetBeans IDE 18) as shown in Figure-4.1. This application comprises key processes such as Purchase Order, Issue Order, Item Creation, Transfer Order etc, and it's exhibiting the functionalities of SecureERP. We have configured Oracle database 21c as a back-end server, we deployed Hyperledger Fabric Network on Ubuntu VM. We used Ngrok for integration of Front-end (SecureERP) and Hyperledger Fabric Network. Moreover, to enhance performance of our REST APIs, we used Redis Sever, deployed along Hyperledger Fabric.

Eventually, we created a test-case that validates our complete POC process, beginning with the integration and extending to validate and verify transaction integrity. The test-case also ensures the synchronization of transactional data in both the systems (SecureERP and Fabric Blockchain). In the next section, we will start the POC using the created test-case in a stepwise approach, ensuring a systematic and thorough evaluation of the system.

5.2 Test-Case

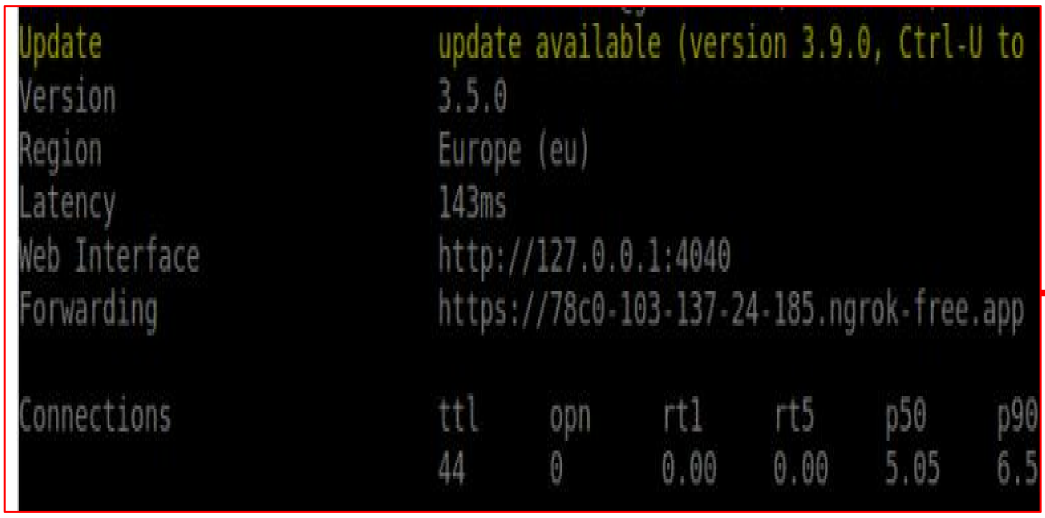
Our Front-end application includes multiple forms such Purchase Order, Item Creation, and Issue Order etc. However, we conduct our POC on "**Item Creation Form**" in this paper. The steps involved in Test-Case process, are discussed in following sub-sections and screenshots or images of each step are also provided as a result.

5.2.1 Step -1 : Integration of SecureERP and Hyperledger Fabric

As stated earlier in the implementation section (chapter 4), we have created three VM Servers. Our Front-end Application is hosted on separate VM, while the Hyperledger Fabric Blockchain is deployed and operating at another VM. In order to integrate Front-end application with Hyperledger Fabric, we used Ngrok Server. The Ngrok is deployed and running along Hyperledger Fabric, it gives us a URL or path that can be used to access our blockchain platform form Front-end application.

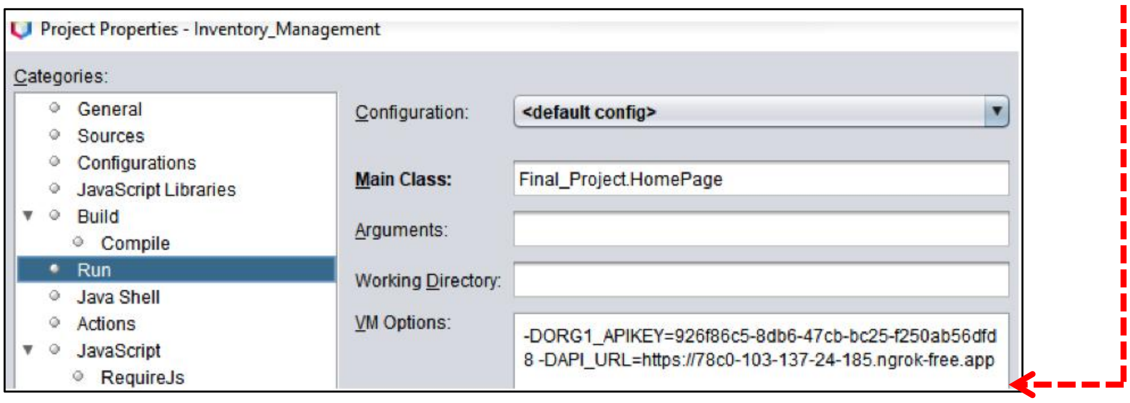
We integrated our Front-end application and Hyperledger Fabric using Ngrok. Ngrok gave a URL (**https://78c0-103-137-24-185.ngrok-free.app**), shown in Figure-5.1. We have provided this URL in the properties section of our Front-end application or project in NetBeans as shown at Figure-5.2. Doing so, both the systems will be integrated. Now, transactions done at SecureERP's database will also be sent to Hyperledger Fabric blockchain according to our use-case. A complete transaction cycle and its execution within both the systems are demonstrated in succeeding sections.

Integration - SecureERP and Fabric Network



```
Update                          update available (version 3.9.0, Ctrl-U to
Version                         3.5.0
Region                          Europe (eu)
Latency                         143ms
Web Interface                   http://127.0.0.1:4040
Forwarding                      https://78c0-103-137-24-185.ngrok-free.app

Connections                     ttl     opn     rt1     rt5     p50     p90
                                44      0       0.00    0.00    5.05    6.5
```

Figure -5.1



Project Properties - Inventory_Management

Categories:

- General
- Sources
- Configurations
- JavaScript Libraries
- Build
  - Compile
- Run
- Java Shell
- Actions
- JavaScript
  - RequireJs

Configuration:	<default config>
Main Class:	Final_Project.HomePage
Arguments:	
Working Directory:	
VM Options:	-DORG1_APIKEY=926f86c5-8db6-47cb-bc25-f250ab56dfd8 -DAPI_URL=https://78c0-103-137-24-185.ngrok-free.app

Figure- 5.2

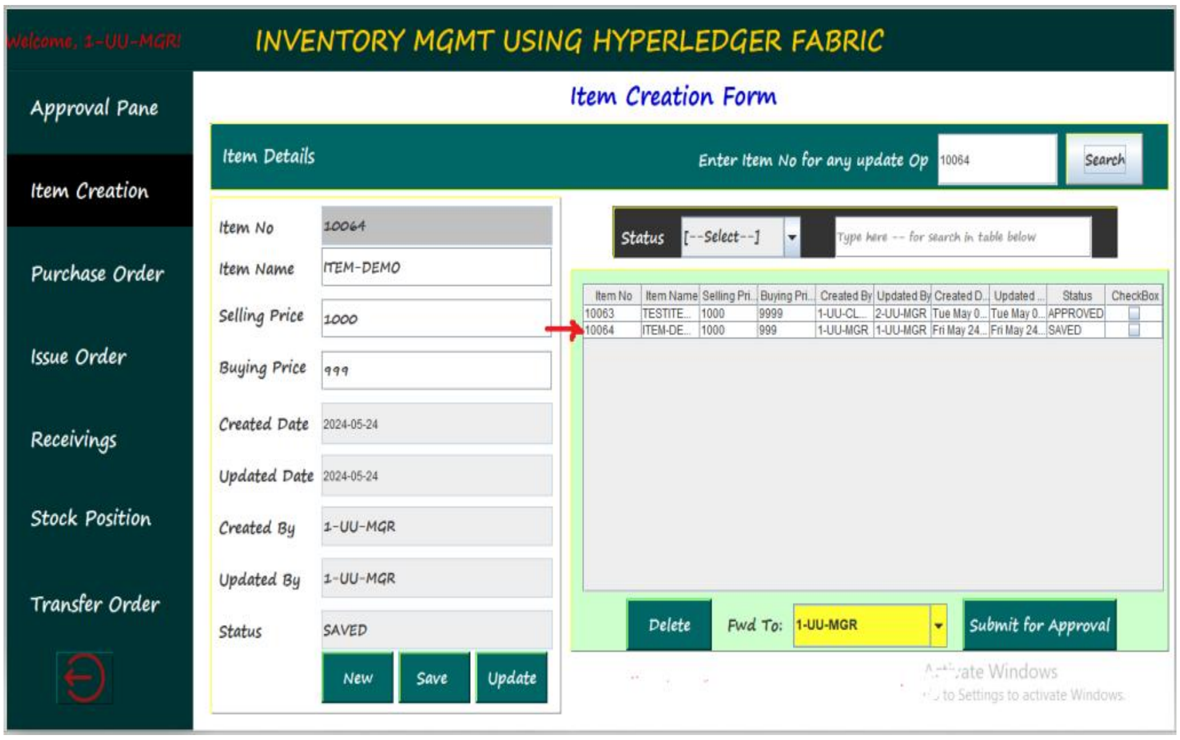
5.2.2 Step -2 : Initiating Transaction

The second step in our test case involves making a transaction in the SecureERP system. We used the "Item Creation Form" in the front-end application. After logging into the application, we created a new item record with Item no = **10064** and Item Name = "**ITEM-DEMO**", as illustrated in Figure 5.3. Once the record is saved by clicking on the "**Save**" button, it is stored in both the systems i.e. the SecureERP's Oracle Database, as shown in Figure 5.4, and the Hyperledger Fabric blockchain, as depicted in Figure 5.5.

This successful integration of both the systems validates the desired functionality of our REST APIs and design concept. The ***POST HTTP*** method (REST API) associated with the "Save" button is used to send the payload to the Hyperledger Fabric blockchain.

This integration ensures that the transactions are consistently recorded in both the systems, maintaining transaction integrity and enhancing the system's reliability.

Item Creation or Transaction in SecureERP : Front-end Application



Enter Item No for any update Op: 10064

Item No	10064
Item Name	ITEM-DEMO
Selling Price	1000
Buying Price	999
Created Date	2024-05-24
Updated Date	2024-05-24
Created By	1-UU-MGR
Updated By	1-UU-MGR
Status	SAVED

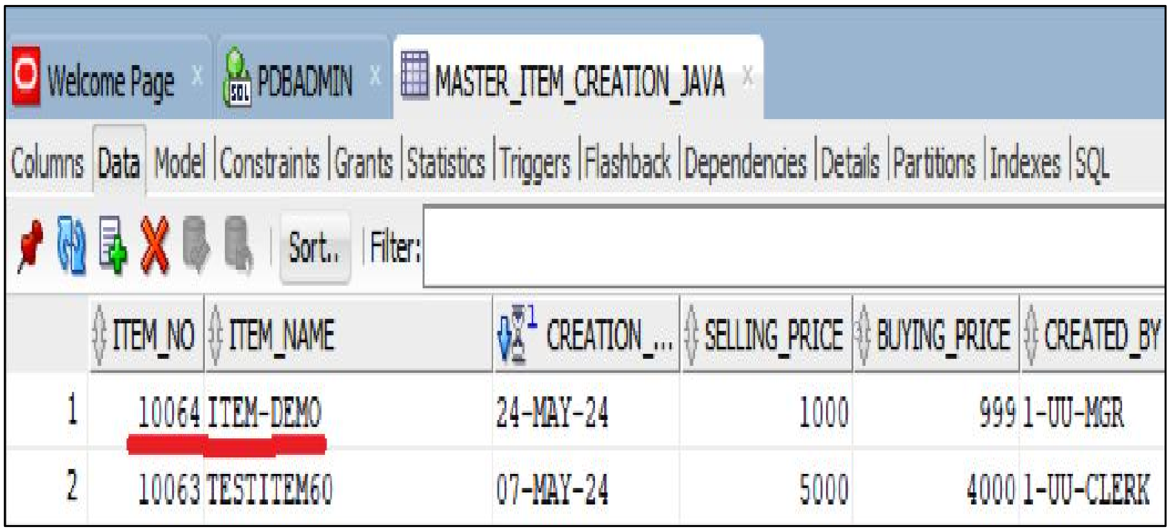
Status: [--Select--] Type here -- for search in table below

Item No	Item Name	Selling Pri...	Buying Pri...	Created By	Updated By	Created D...	Updated ...	Status	CheckBox
10063	TESTITE...	1000	9999	1-UU-CL...	2-UU-MGR	Tue May 0...	Tue May 0...	APPROVED	☐
10064	ITEM-DE...	1000	999	1-UU-MGR	1-UU-MGR	Fri May 24...	Fri May 24...	SAVED	☐

Fwd To: 1-UU-MGR

Figure 5.3

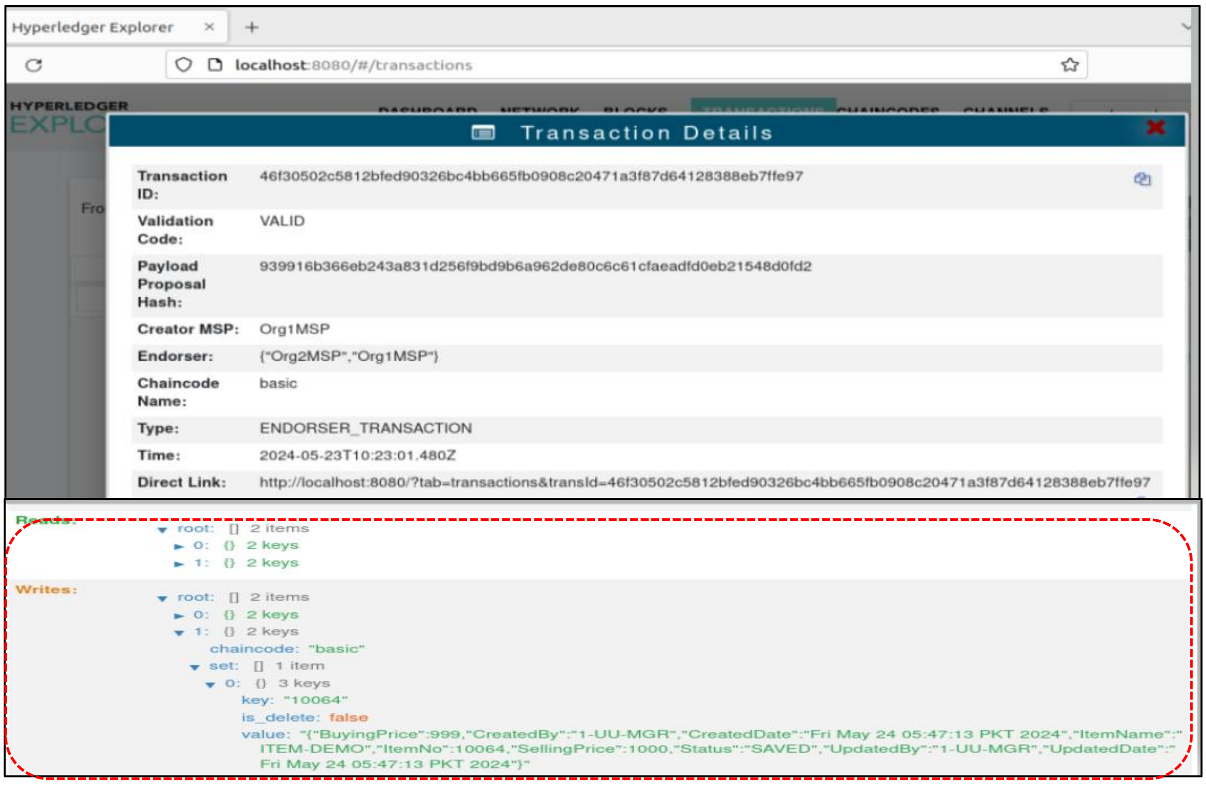
Transaction (Item Record) - Oracle Database (SecureERP System)



	ITEM_NO	ITEM_NAME	CREATION_...	SELLING_PRICE	BUYING_PRICE	CREATED_BY
1	10064	ITEM-DEMO	24-MAY-24	1000	999	1-UU-MGR
2	10063	TESTITEM60	07-MAY-24	5000	4000	1-UU-CLERK

Figure 5.4

<u>Transaction (Item Record) - Blockchain (Hyperledger Fabric)</u>



Transaction ID:	46f30502c5812bfed90326bc4bb665fb0908c20471a3f87d64128388eb7ffe97
Validation Code:	VALID
Payload Proposal Hash:	939916b366eb243a831d256f9bd9b6a962de80c6c61cfaeadfd0eb21548d0fd2
Creator MSP:	Org1MSP
Endorser:	{"Org2MSP","Org1MSP"}
Chaincode Name:	basic
Type:	ENDORSER_TRANSACTION
Time:	2024-05-23T10:23:01.480Z
Direct Link:	http://localhost:8080/?tab=transactions&transId=46f30502c5812bfed90326bc4bb665fb0908c20471a3f87d64128388eb7ffe97

```
Reads:      ▾ root:  [] 2 items
              ▸ 0:  {} 2 keys
              ▸ 1:  {} 2 keys
Writes:     ▾ root:  [] 2 items
              ▸ 0:  {} 2 keys
              ▾ 1:  {} 2 keys
                  chaincode: "basic"
                ▾ set:  [] 1 item
                  ▾ 0:  {} 3 keys
                      key: "10064"
                      is_delete: false
                      value: "{"BuyingPrice":999,"CreatedBy":"1-UU-MGR","CreatedDate":"Fri May 24 05:47:13 PKT 2024","ItemName":"
                        ITEM-DEMO","ItemNo":10064,"SellingPrice":1000,"Status":"SAVED","UpdatedBy":"1-UU-MGR","UpdatedDate":"
                        Fri May 24 05:47:13 PKT 2024"}"
```

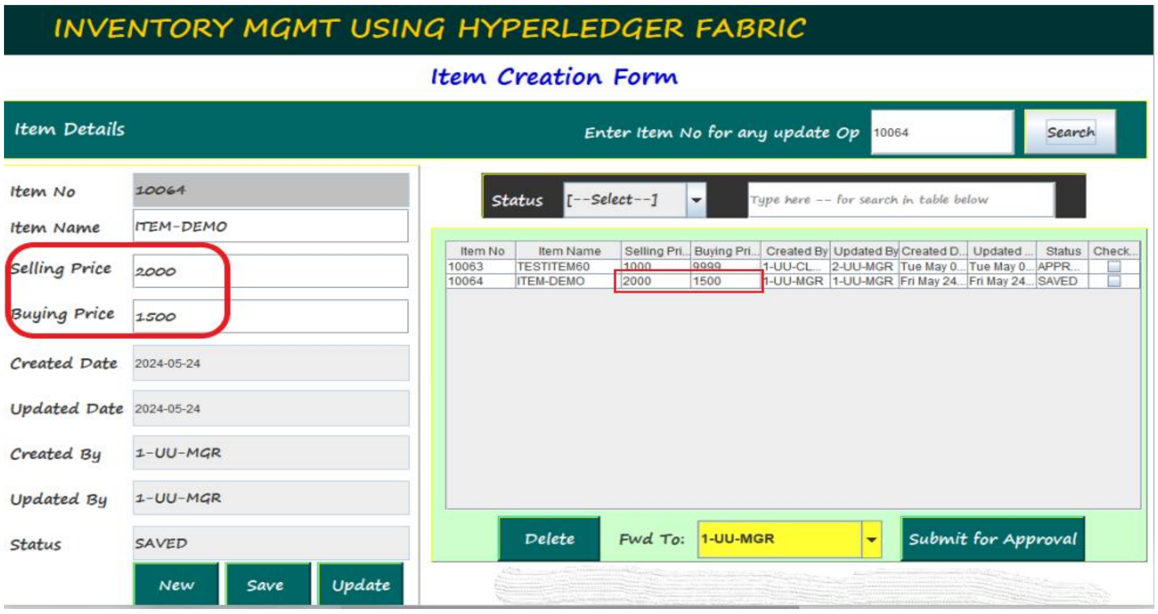
Figure- 5.5

5.2.3 Step -3 : Updating Transaction

In step-2, we have seen that transaction or record is successfully created in both the systems. This proved that both SecureERP and Hyperledger Fabric are seamlessly integrated, and the REST APIs are also working according to our use-case. In our test-case, step-3 represents the validation or testing of update operation in both the systems. It is necessary to mention that, transactions or data in the Hyperledger Fabric blockchain are immutable, which cannot be updated or deleted. Instead, updating existing record results in creation of new transaction at blockchain level (Hyperledger Fabric).

The new record will be linked to original transaction in order to maintain the history or transaction trail for audit purposes. Maintaining an immutable history of transactions by Hyperledger Fabric will ensure the transaction integrity and security in SecureERP system. To further explain this, let's consider a scenario, where we update the Selling and Buying prices of Item with Item no =10064. We changed the Selling Price to **‘2000’** and Buying Price to **‘1500’,** as shown at Figure-5.6. The transaction is updated at Oracle database as depicted at Figure-5.7, whereas, a new transaction at blockchain level is also created that captures the details for this update operation on same Item no=10064, as shown at Figure-5.8. This demonstrates that we have successfully tested the update operation in our proposed model, aligning with our requirements.

Updating Transaction / Record - SecureERP Application


INVENTORY MGMT USING HYPERLEDGER FABRIC
Item Creation Form
Item Details
Enter Item No for any update Op
Search
Status
[--Select--]
Type here -- for search in table below
Delete
Fwd To:
Submit for Approval
New
Save
Update

Enter Item No for any update Op: 10064

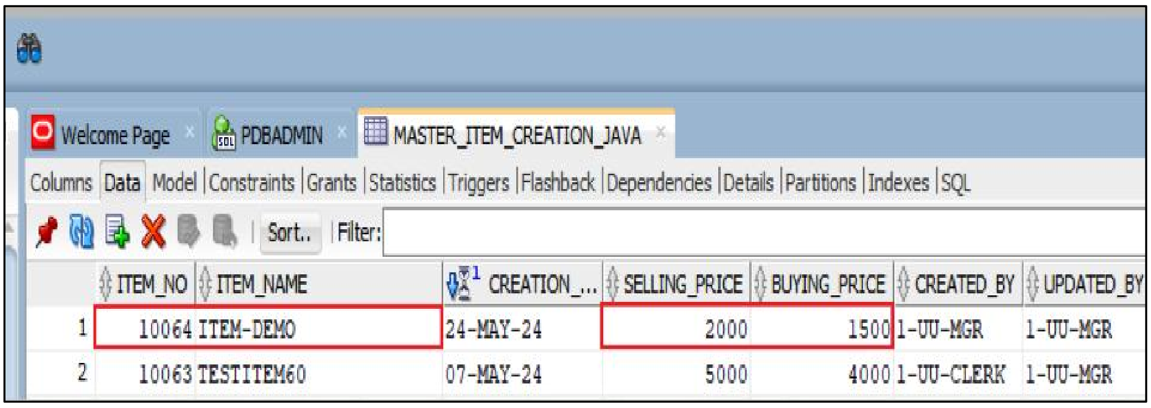
Item No	10064
Item Name	ITEM-DEMO
Selling Price	2000
Buying Price	1500
Created Date	2024-05-24
Updated Date	2024-05-24
Created By	1-UU-MGR
Updated By	1-UU-MGR
Status	SAVED

Item No	Item Name	Selling Pri...	Buying Pri...	Created By	Updated By	Created D...	Updated ...	Status	Check...
10063	TESTITEM60	1000	9999	1-UU-CL...	2-UU-MGR	Tue May 0...	Tue May 0...	APPR...	☐
10064	ITEM-DEMO	2000	1500	1-UU-MGR	1-UU-MGR	Fri May 24...	Fri May 24...	SAVED	☐

Fwd To: 1-UU-MGR

Figure- 5.6

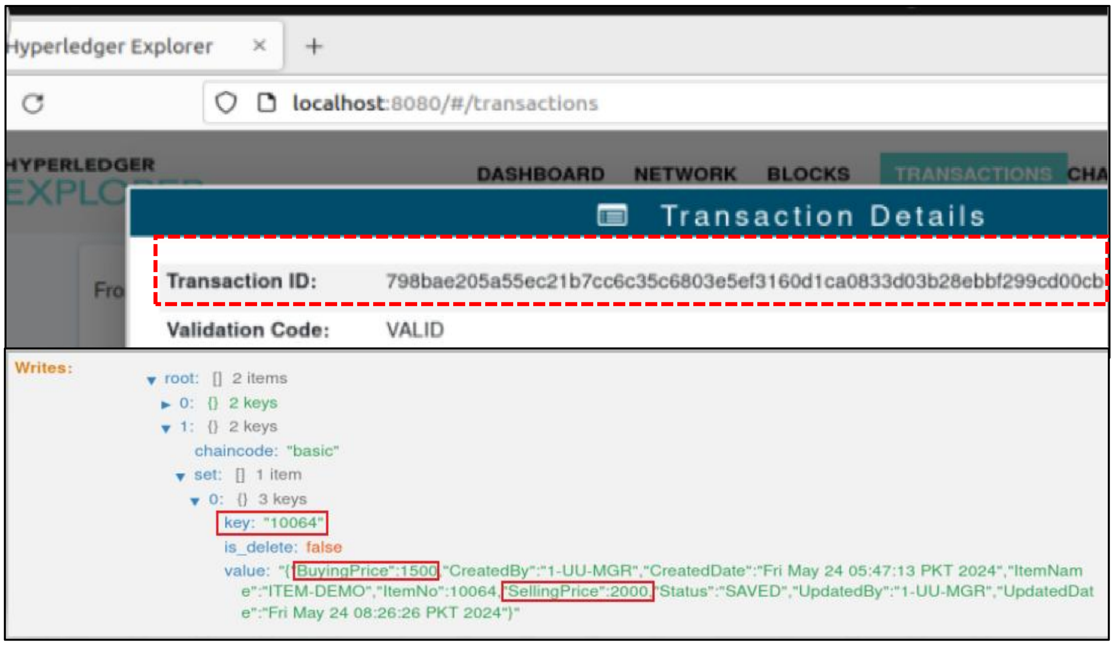
Updated Transaction / Record - SecureERP’s Database (Oracle)



	ITEM_NO	ITEM_NAME	CREATION_...	SELLING_PRICE	BUYING_PRICE	CREATED_BY	UPDATED_BY
1	10064	ITEM-DEMO	24-MAY-24	2000	1500	1-UU-MGR	1-UU-MGR
2	10063	TESTITEM60	07-MAY-24	5000	4000	1-UU-CLERK	1-UU-MGR

Figure - 5.7

<u>Updated Transaction / Record - Blockchain (Hyperledger Fabric)</u>



```
Transaction ID:      798bae205a55ec21b7cc6c35c6803e5ef3160d1ca0833d03b28ebbf299cd00cb
Validation Code:     VALID
Writes:
        ▾ root:  []  2 items
          ▸ 0:  {}  2 keys
          ▾ 1:  {}  2 keys
              chaincode: "basic"
            ▾ set:  []  1 item
              ▾ 0:  {}  3 keys
                  key: "10064"
                  is_delete: false
                  value: "{"BuyingPrice":1500,"CreatedBy":"1-UU-MGR","CreatedDate":"Fri May 24 05:47:13 PKT 2024","ItemNam
                    e":"ITEM-DEMO","ItemNo":10064,"SellingPrice":2000,"Status":"SAVED","UpdatedBy":"1-UU-MGR","UpdatedDat
                    e":"Fri May 24 08:26:26 PKT 2024"}"
```

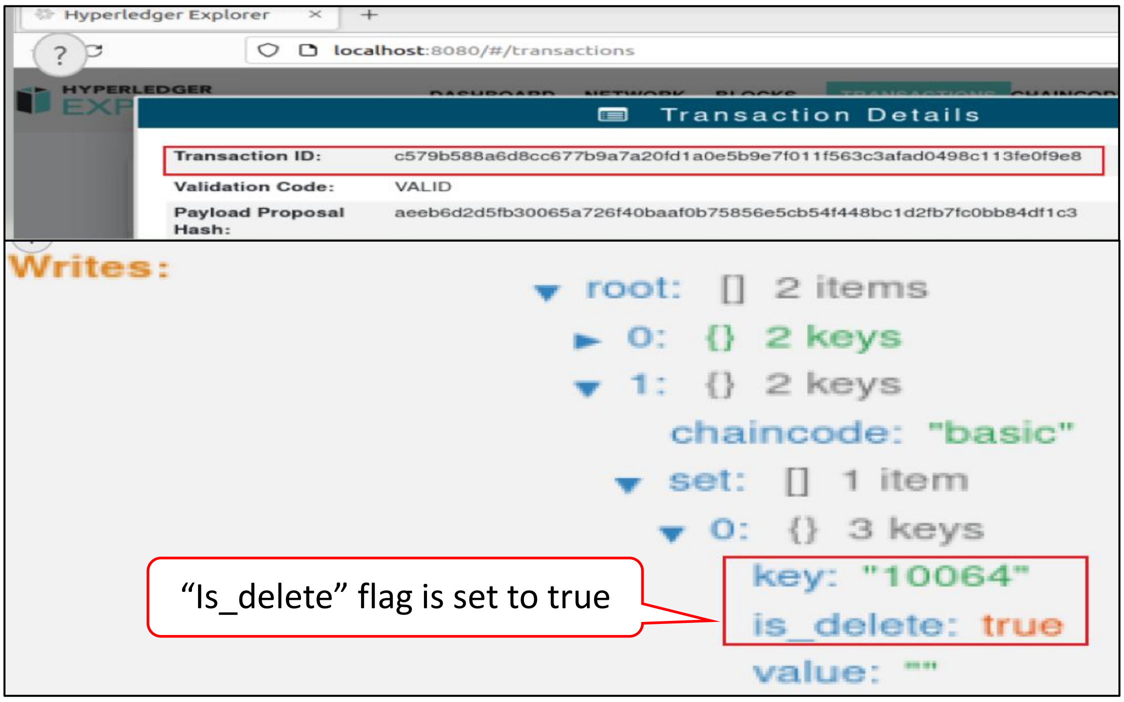
Figure 5.8

5.2.4 Step -4 : Deleting Transaction

This is crucial step, since records in the sOracle database or SecureERP can be be deleted or manipulated. In-case, a record is deleted in SecureERP, then a new transaction will be generated in the blockchain that will capture the details of this deleted record. As an example, we have deleted a transaction having ***item no 10064*** from SecureERP. However, new transaction is created at Blockchain level, as shown at Figure -5.9.

Usually, for deleted transaction in the Hyperledger Fabric Blockchain, a "**is_delete**" flag is set to "**true**", which means that this record or transaction is deleted from state database (CouchDB) and also from SecureERP Database (Oracle Database), but the transaction history is maintained at blockchain level. This will ensure security and integrity of the transactions. This action is done through DELETE method in our REST API.

Deleted Transaction


Hyperledger Explorer
localhost:8080/#/transactions
HYPERLEDGER
Transaction Details

Transaction ID: c579b588a6d8cc677b9a7a20fd1a0e5b9e7f011f563c3afad0498c113fe0f9e8
Validation Code: VALID
Payload Proposal Hash: aeeb6d2d5fb30065a726f40baaf0b75856e5cb54f448bc1d2fb7fc0bb84df1c3

Writes:

```
▼ root: [] 2 items
  ► 0: {} 2 keys
  ▼ 1: {} 2 keys
      chaincode: "basic"
    ▼ set: [] 1 item
      ▼ 0: {} 3 keys
          key: "10064"
          is_delete: true
          value: ""
```

"Is_delete" flag is set to true

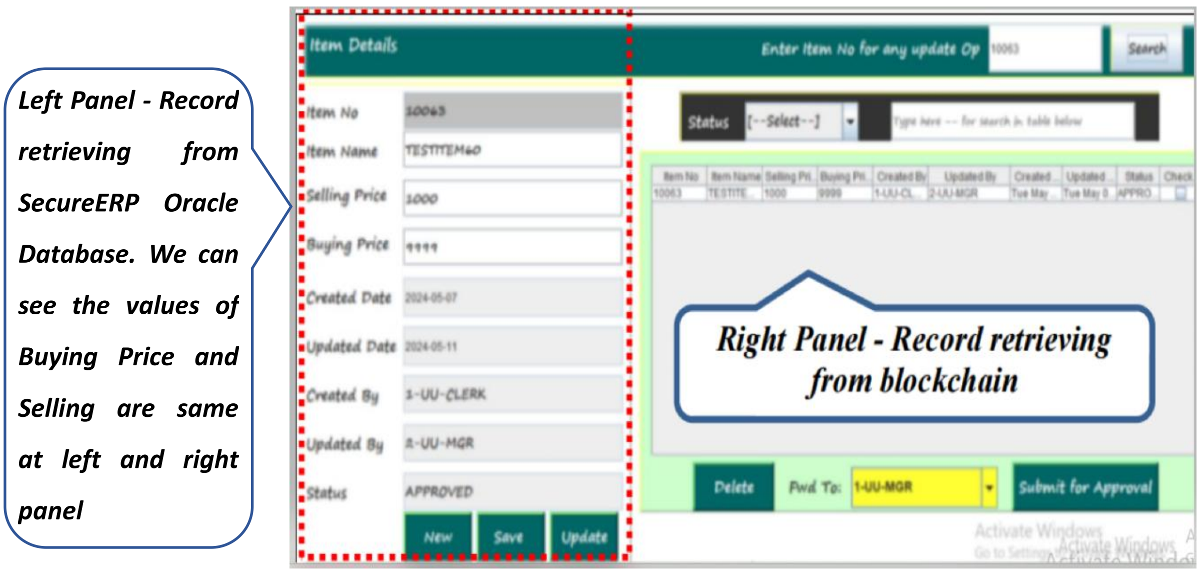
Figure - 5.9

5.2.5 Step 5 - Alerts Generation for Illegitimate Manipulation

The data and transactions in the SecureERP Oracle database are vulnerable to changes, the administrators with broad access privileges can manipulate data. In order to restrict administrators from such type of an illegitimate manipulation, we have implemented a comprehensive alert generation mechanism. In this mechanism, transactions in both the systems are compared, if any discrepancies found, our model will generate an alert about that particular transaction. This alert mechanism will ensure quick notification about illegitimate manipulation, thus ensuring the integrity of transaction in the SecureERP. We validated the synchronization and integrity of transaction using following steps:-

- **Initial State.** As depicted at Figure-5.10, initially transactional data are similar in both the systems. On the left-panel, data is retrieved from Oracle database (SecureERP) and on the right-panel data is retrieved from Hyperledger Fabric blockchain.

Initial Status


Left Panel - Record retrieving from SecureERP Oracle Database. We can see the values of Buying Price and Selling are same at left and right panel
Right Panel - Record retrieving from blockchain
Item Details
Enter Item No for any update Op
Search
Status
[--Select--]
Type here -- for search in table below
New
Save
Update
Delete
Submit for Approval

Item No	10063
Item Name	TESTITEM60
Selling Price	1000
Buying Price	9999
Created Date	2024-05-07
Updated Date	2024-05-11
Created By	1-UU-CLERK
Updated By	2-UU-MGR
Status	APPROVED

Enter Item No for any update Op: 10063

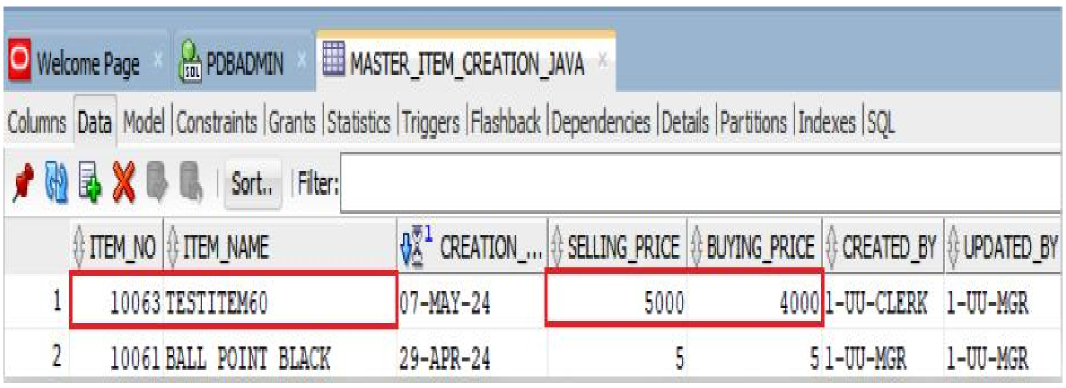
Item No	Item Name	Selling Pri...	Buying Pri...	Created By	Updated By	Created ...	Updated ...	Status	Check
10063	TESTITE...	1000	9999	1-UU-CL...	2-UU-MGR	Tue May ...	Tue May 0...	APPRO...	

Fwd To: 1-UU-MGR

Figure-5.10

- **Illegitimate Modifications**. In our use-case, transaction once approved cannot be changed in both the systems. Since, Oracle database is vulnerable to unauthorized manipulation. Therefore, as a test case, we intentionally manipulated the approved transaction with **Item no 10063,** and updated the **Selling Price** to **5000** and **Buying Price** to **4000,** by directly accessing our Oracle database. This action is shown at Figure-5.11. In next step, we will see, our model detects this manipulation and generated an alert.

Illegitimate Manipulation in Oracle Database

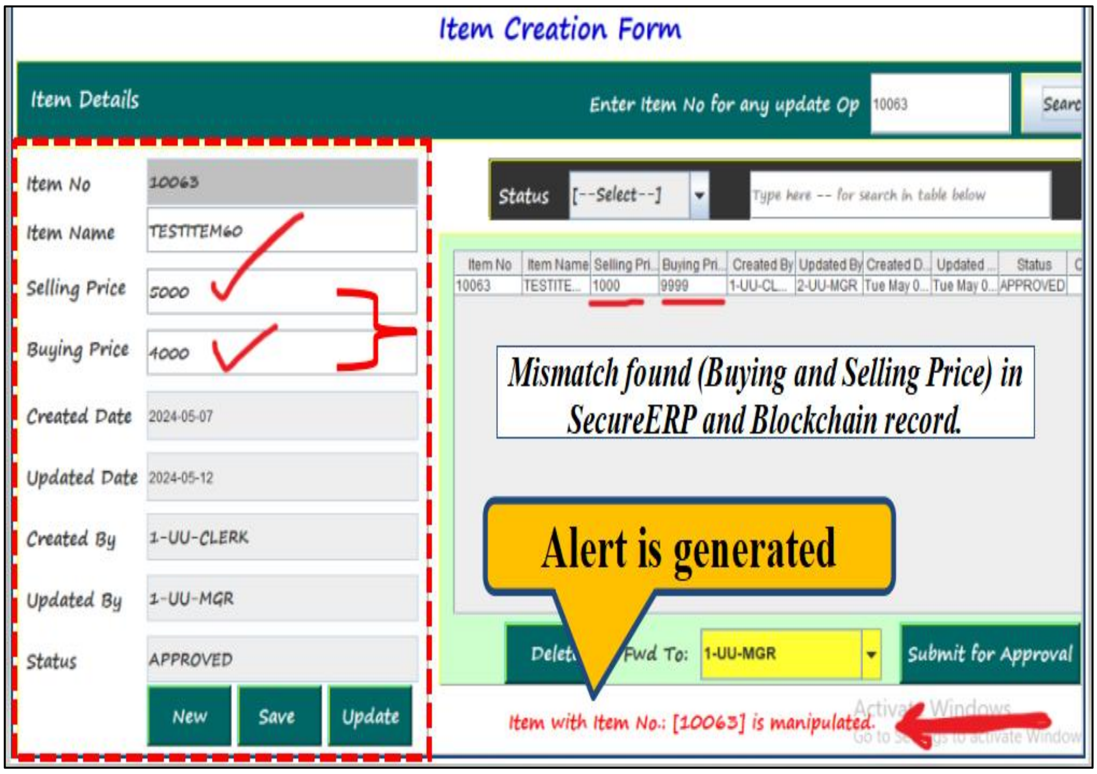


	ITEM_NO	ITEM_NAME	CREATION_...	SELLING_PRICE	BUYING_PRICE	CREATED_BY	UPDATED_BY
1	10063	TESTITEM60	07-MAY-24	5000	4000	1-UU-CLERK	1-UU-MGR
2	10061	BALL POINT BLACK	29-APR-24	5	5	1-UU-MGR	1-UU-MGR

Figure-5.11

- **Alerts Generation**. After changing record or transaction in the SecureERP's database as shown in Figure-5.11, inconsistency exists between data stored in ERP and blockchain systems, therefore, our model generates an "**Alert**" as shown in Figure-5.12. With this, we concluded our POC, and we have tested and validated all the relevant scenarios regarding transaction integrity, security and synchronization of data between both the systems. The response of our model was efficient and according to design conception.

<u>Alert Generated</u>



Item Creation Form

Item Details

Enter Item No for any update Op: 10063

Item No	10063
Item Name	TESTITEM60
Selling Price	5000
Buying Price	4000
Created Date	2024-05-07
Updated Date	2024-05-12
Created By	1-UU-CLERK
Updated By	1-UU-MGR
Status	APPROVED

Status: [--Select--]

Type here -- for search in table below

Item No	Item Name	Selling Pri...	Buying Pri...	Created By	Updated By	Created D...	Updated ...	Status
10063	TESTITE...	1000	9999	1-UU-CL...	2-UU-MGR	Tue May 0...	Tue May 0...	APPROVED

Mismatch found (Buying and Selling Price) in SecureERP and Blockchain record.

Alert is generated

Fwd To: 1-UU-MGR

Item with Item No.: [10063] is manipulated.

Figure-5.12

5.3 Summary

In this chapter, we presented the Proof-of-Concept (POC) and the results of our proposed model, with each step in the POC process accompanied by a figure or screenshot to validate the outcomes. We created a test-case to validate the complete execution cycle, from integration to synchronization of transactions within the systems. The results were aligned with our design criteria, demonstrating that the model performs effectively, meets the specified requirements, and proves the feasibility of our approach.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Overview

In this section, we discuss the feasibility, adoption potential of our proposed model. We also highlight and discuss key challenges that are encountered during integration phase. The purpose is to provide a road-map to those enterprises, which desire to induct blockchain like technologies into their systems.

6.2 Feasibility of our Proposed Model

Our model has potential to fulfills the requirements, and mitigates the risks associated with the transactions integrity like problems within SecureERP system. We have presented two scenarios for integration of SecureERP with blockchain. By leveraging the capabilities of Hyperledger Fabric and adopting Raft-based consensus mechanism, we believe that our model is not only capable to address the current challenges, but also aligns with their future scenario (**Scenario-1 , discussed in the design section**).

6.3 Adoption Potential of Our Model

Our model primarily focusing on ensuring transactions integrity management and security in SecureERP application. However, it is adaptable and can be used by an enterprises, which are looking to operate and communicate over Permissioned blockchain platforms. Our model can be adopted by organizations in various sectors, such as, Supply chain, healthcare specifically in Hospitals Management System, education, Excise & Taxation. Our proposed model provides a solid framework for enterprises, who want to bring trust, transparency and integrity in their operations through blockchain.

6.4 Challenges or Limitations

During POC phase, it has been learnt that Hyperledger Fabric Blockchain boasts their own merits, meanwhile integrating Blockchain with existing ERP (Centralized System) will definitely increase complexity by introducing a completely new technology stack into their IT environment just for the sake of transaction's integrity.

Former researchers have emphasized the importance of a compelling use-case for blockchain adoption. However, inherent challenges such as complexity, latency, hardware, and development costs contribute to a high failure rate for blockchain projects [50]. The challenges discussed below are not exclusive to our proposed model, rather, these are the common obstacles that enterprises may face when integrating ERP like applications with Permissioned blockchain platform. The critical challenges are:-

- **Complexity**. Like other technologies, Blockchain also offers many advantages, certainly trade-off exists. The most critical challenges in the private or Permissioned blockchain is it's inherent complexity. The architecture of blockchain is very complex, which is correctly described by International Data Corporation in its report "***Barriers to Blockchain Adoption in Europe, March 2020, Doc:EUR14573320***" [48]. This report says "Complexity is the major factor that increases failure rate/ risk of blockchain project". In our use-case, SecureERP is operating under controlled environment having complex architecture, so, integrating with Hyperledger Fabric or any other Permissioned blockchain will further increase its complexity.

- **Cost.** Integrating SecureERP with Hyperledger Fabric, will increase the cost significantly. These costs may include additional hardware resources and development efforts required to maintain the Blockchain infrastructure.

- **Business Use-case**. The concept presented by Mohammad Jabbed Morshed Chowdhury et al [1], is supported. Organizations must analyze their use-case(s) before implementing blockchain. If use-case is not properly analyzed and evaluated, then integrating blockchain into exiting IT environment would not justify the potential benefits.

- **Transformation/ Alteration at existing architecture.** Transaction in the blockchain involves many phases such as Endorsement, Consensus and Chaincode etc.

Integrating Blockchain and ERP will require major alterations at existing ERP application to operate over blockchain environment. Moreover, it may require changes in the business-process and how business interact with customers and partners.

- **Latency and Low TPR.** Usually, ERP systems offer very high TPR, because of having centralized database. However, integrating these ERPs with blockchain will increase latency as well as reduce TPR, depends on use-case and synchronization mechanism adopted between the systems.

6.5 Future Work

Although, integrating Hyperledger Fabric with existing ERPs application can address key challenges (such as privacy, transparency, trust, immutability, transactional integrity and point of single failure), but this integration comes with a huge costs in terms of hardware and development efforts. There are enterprises, where data confidentiality is critical, and usually prefers over data transparency. Some times an enterprises may not require a complete blockchain platform to stack into their complex IT infrastructure. Instead, these organizations are looking for an immutable and temper-proof ledger. Generally these organizations have well-established Disaster Recovery (DR) sites, and do not want to shift from centralized to decentralized (Blockchain) architecture. In the light of these considerations, we suggest that a custom blockchain-like solution may be developed. This custom-blockchain should provide basic functionalities such as immutable transactions records, distributed transactions support, and utilizing SHA for security and hashing purposes. Moreover, an enterprises where Oracle-based ERP (eBS) system is deployed, such enterprises must leverage the native and innovative technology of "Immutable" and "Blockchain" tables, provided by Oracle in its latest databases (21c onward). This may offer an alternative approach to enterprises, to bring blockchain like technology into their environment without tears.

6.6 Conclusion

We have demonstrated a model and framework that integrates existing Oracle-based ERP systems (aka SecureERP) with Hyperledger Fabric. In order to test our proposed model, we carried out Proof-of-Concept through utilizing relevant technologies. Generally, ERP systems have higher transaction processing rates (TPR) compared to blockchain. SecureERP is a robust system having daily transactional volume range between **10000** to **15000**. We used Raft-based consensus mechanism in our proposed model to align with this high TPR of SecureERP. In addition, we also discussed that our model is not restricted to SecureERP only, rather, it can be adopted by any enterprise who wants to integrate Permissioned blockchain in their environment. Our model effectively addresses key issues within SecureERP regarding transaction integrity, and security. During implementation of our model, we learnt that analyzing use-case for blockchain is critical, without analyzing use-case, potential benefits couldn't be achieved. We have highlighted few obstacles that enterprises may encounter during the integration process. In-fact, Blockchain is a leading technology, however, the complexities and costs involved in the blockchain-based projects often contribute to their failure. Moreover, it is not necessary that blockchain could be stacked at every project, rather depends on use-case. At the end, we recommend a future work to be carried out on custom-based blockchain solutions for enterprises, where data confidentiality is critical and prefers over data transparency

CHAPTER 7

REFERENCES

REFERENCE

[1] Mohammad Jabbed Morshed Chowdhury (2018) et al. Blockchain versus Database: A Critical Analysis. Available at: https://www.researchgate.net/publication/327483781

[2] Tinal Parikh, The ERP of the Future: Blockchain of Things. Online ISSN : 2394-4099

[3] Tehreem Aslam, Ayesha Maqbool et al. Blockchain Based Enhanced ERP Transaction Integrity Architecture and PoET Consensus. DOI:10.32604/cmc.2022.019416

[4] LAHLOU Imane, MOTAKI. "Integrating Blockchain with ERP systems for better supply chain performance" DOI: 10.1109/LOGISTIQUA55056.2022.9938086

[5] Abdelhak Belhi, Houssem Gasmi et al. Integration of Business Applications with the Blockchain: Odoo and Hyperledger Fabric Open Source Proof of Concept. Available online at www.sciencedirect.com

[6] Kok Yong Chan, Johari Abdullah et al. A Framework for Traceable and Transparent Supply Chain Management for Agri-food Sector in Malaysia using Blockchain Technology

[7] A. Faccia and P. Petratos, "Blockchain, Enterprise Resource Planning (ERP) and Accounting Information Systems (AIS): Research on eProcurement and System Integration, https://doi.org/10.3390/app11156792

[8] A R Komala, I Gunanda "Development of Enterprise Resource Planning using Blockchain" doi:10.1088/1757-899X/879/1/012141

[9] Lahlou Imane et al. "Towards Blockchain-Integrated Enterprise Resource Planning:A Pre-Implementation Guide" https://doi.org/10.3390/computers13010011.

[10] https://www.hyperledger.org/

[11] MUHAMMAD IMRAN et al, Data Vaults for Blockchain-Empowered Accounting Information Systems - DOI 10.1109/ACCESS.2021.3107484.

[12] Artem Barger et al, Orion: A Centralized Blockchain Database with Multi-Party Data Access Control 2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)|979-8-3503-1019-1/23/$31.00©2023IEEE| DOI: 10.1109/ICBC56567.2023.10174914

[13] Arman Raj et al, Enhancing Security Feature in Financial Transactions using Multi chain Based Blockchain Technology 2023 4th International Conference on Intelligent Engineering and Management (ICIEM) | 979-8-3503-4112-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICIEM59379.2023.10166589

[14] Karthik H P et al, Blockchain Technology: Converting Data Integrity and Security in Supply Chain Management. 2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON) | 979-8-3503-8247-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/UPCON59197.2023.10434412.

[15] J. R. Muscatello, et al. "Implementing enterprise resource planning (ERP) systems in small and midsize manufacturing firms," Int Jrnl of Op & Prod Management, vol. 23, no. 8, pp. 850–871, Aug. 2003.

[16] L. Sislian and A. Jaegler, "Linkage of blockchain to enterprise resource planning systems for improving sustainable performance," Business Strategy and the Environment, Oct. 2021.

[17] https://blogs.oracle.com/blockchain/post/blockchain-tables-in-oracle-database-technology-convergence

[18] M. Aloqaily, A. Boukerche, O. Bouachir, F. Khalid, and S. Jangsher, "An Energy Trade Framework Using Smart Contracts: Overview and Challenges," IEEE Network, vol. 34, no. 4, pp. 119–125, Jul. 2020

[19] https://blogs.oracle.com/blockchain/post/you-too-can-quickly-build-a-blockchain-poc-using-pre-assembled-oracle-cloud-tools.

[20] Ruwanthi Perera, et al. Role of Blockchain Technology in ERP Systems for a Transparent Supply Chain: A Systematic Review of Literature DOI: 10.1109/ICARC57651.2023.10145618

[21] Xaysomphone Thipphonexai, Yan Guanghui. Research on analysis and design of cloud ERP based on blockchain technology DOI: 10.1109/ICVRIS51417.2020.0019

[22] A R Komala, Gunanda. Development of Enterprise Resource Planning using Blockchain. doi:10.1088/1757-899X/879/1/012141.

[23] https://www.techtarget.com/searchoracle/definition/Oracle-E-Business-Suite.

[24] Dr. Ronald et al. Challenges and Opportunities of Using Blockchain in Supply Chain Management. Global Business and Management Research: An International Journal Vol. 13, No. 3 (2021)

[25] Moutaz Haddara, et al. Enterprise Systems and Blockchain Technology: The Dormant Potentials. DOI link: https://doi.org/10.1016/j.procs.2021.01.203

[26] Thomas Kitsantas. Exploring Blockchain Technology and Enterprise Resource Planning System: Business and Technical Aspects, Current Problems, and Future Perspectives. https://doi.org/10.3390/su14137633

[27] Pratik Thantharate, Anurag Thantharate. ZeroTrustBlock: Enhancing Security, Privacy, and Interoperability of Sensitive Data through ZeroTrust Permissioned Blockchain DOI: https://doi.org/10.3390/bdcc7040165

[28] Glory Ugochi Ebirim et al. A CRITICAL REVIEW OF ERP SYSTEMS IMPLEMENTATION IN MULTINATIONAL CORPORATIONS: TRENDS, CHALLENGES, AND FUTURE DIRECTIONS.
DOI: 10.51594/ijmer.v6i2.770

[29] Saveen A ,et al. Blockchain Ready Manufacturing Supply Chain Using Distributed Ledger. Available at https://www.researchgate.net/publication/308163874_Blockchain_Ready_Manufacturing_Supply_Chain_Using_Distributed_Ledger?enrichId=rgreq-2bb3759b9e6c21eceecfd1a2dab6a32f-XXX&enrichSource=Y292ZXJQYWdlOzMwODE2Mzg3NDtBUzo0MTYzNDQ1MjU4MTk5MDlAMTQ3NjI3NTY3ODQ1Mw%3D%3D&el=1_x_3

[30] Madhavi Vinayak Godbole. REVOLUTIONIZING FINANCIAL LANDSCAPES: THE INTERPLAY OF AI, ML, ERP, AND ORACLE IN DIGITAL TRANSFORMATION. Available at. DOI : https://www.doi.org/10.56726/IRJMETS49100

[31] Malyavkina L.I. et al. Blockchain technology as the basis for digital transformation of the supply chain management system: benefits and implementation challenges. Available at http://creativecommons.org/licenses/by-nc/4.0/)

[32] UDIT AGARWAL et al. Blockchain_Technology_for_Secure_Supply_Chain_Management_A_Comprehensive_Review. DOI :10.1109/ACCESS.2022.3194319

[33] Nejc Rožmana, Rok Vrabiča et al. Distributed logistics platform based on Blockchain and IoT. Available online at www.sciencedirect.com

[34] Thomas Kitsantas. Exploring Blockchain Technology and Enterprise Resource Planning System: Business and Technical Aspects, Current Problems, and Future Perspectives. DOI: https://doi.org/10.3390/su14137633

[35] David Holtkemper, Günther Schuh. Blockchain as Middleware+. Available at https://inria.hal.science/hal-02419194

[36] Karthik Pulipati, Thume Vamshi Krishna. To Enhance Enterprise Resource Planning with Blockchain: Food Supply Chain. DOI: 10.21275/SR22401220948

[37] Muhammad Rizqi Nur, Luqman Hakim, CHALLENGES IN USING BLOCKCHAIN FOR SUPPLY CHAIN MANAGEMENT INFORMATION SYSTEMS.

[38] Patryk Morawiec, Anna Sołtysik-Piorunkiewicz. Cloud Computing, Big Data, and Blockchain Technology Adoption in ERP Implementation Methodology. Available at. https://doi.org/10.3390/su14073714

[39] Srinivasa Rao Gunturu et al. FinTech - Automatic Payment Process in the ERP System. Available at. DOI: 10.14445/22312803/IJCTT-V72I11P116

[40] Oracle, "Oracle," [Online]. Available: https://www.oracle.com. [Accessed: Dec 10, 2023].

[41] KBA AI, "KBA AI," [Online]. Available: https://kba.ai/. [Accessed: Dec. 10, 2023].

[42] Google,"Google"[Online].Available:https://www.google.com/. [Accessed: Sep 15, 2023].

[43] SoftwareMill, "Hyperledger Fabric Cheat Sheet," [Online]. Available: https://softwaremill.com/hyperledger-fabric-cheat-sheet/. [Accessed: Jan. 2, 2024].

[44] Hyperledger Fabric, "Performance," [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.5/performance.html. [Accessed: Jan. 2, 2024].

[45] Hyperledger Fabric, "Gateway," [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/gateway.html. [Accessed: Jan. 21, 2024].

[46] Hyperledger Wiki, "Hyperledger Explorer," [Online]. Available: https://wiki.hyperledger.org/display/explorer. [Accessed: Apr. 21, 2024].

[47] PubNub, "What is ngrok?," [Online]. Available: https://www.pubnub.com/guides/what-is-ngrok/. [Accessed: Mar 10, 2024].

[48] IDC, "Barriers to Blockchain Adoption in Europe," Mar. 2020, Doc: EUR14573320.

[49] Coinmonks, "Demistify Hyperledger Fabric Components," [Online]. Available: https://medium.com/coinmonks/demistify-hyperledger-fabric-components-87f13325f5d. [Accessed: Sep 21, 2023].

[50] Deloitte, "Global Blockchain Survey," [Online]. Available: https://www2.deloitte.com/xe/en/insights/topics/understanding-blockchain-potential/global-blockchain-survey.html. [Accessed: Apr 20, 2024].

[51] Coinmonks, "Coinmonks," [Online]. Available: https://medium.com/coinmonks/. [Accessed: Feb 5, 2024].