Enhancing Banking Transactions with Blockchain Technology: From Traditional MVC/SOAP API to Hyperledger Fabric.

Mohammad Najm

Final Thesis Report

DECEMBER 2024

DEDICATION

This research is dedicated to all those who have been instrumental in supporting me throughout this journey. To my family, whose love and support have made this work possible, I am deeply grateful. To my mentors, whose guidance and wisdom have been invaluable in shaping my intellectual growth, I offer my sincere gratitude. To all my colleagues and peers who have assisted me in any capacity, I extend my heartfelt thanks. This work is a testament to their unwavering support and encouragement.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my thesis supervisor, Dr. Shashidhara R, for his invaluable support and guidance throughout this endeavor. I also thank the team at Liverpool John Moores University, which is located in Liverpool, United Kingdom, and Team Upgrad Education for their unwavering support throughout my journey. Finally, I extend my heartfelt appreciation to my parents and peers for their encouragement and assistance in helping me complete this projects.

ABSTRACT

Blockchain technology is emerging as an alternative to traditional microservices-based architectures in the financial sector. This study investigates the readiness of Hyperledger Fabric for enterprise adoption in banking, exploring its potential to replace current architectures. Two prototype architectures—one microservices framework and one blockchain framework—are developed to assess applicability in real-world banking. The research addresses knowledge gaps by combining comparative analysis with practical implementation.

It focuses on evaluating trade-offs between traditional and blockchain-based banking infrastructures, offering insights into their viability as replacements. This comprehensive approach aims to identify steps necessary for successful blockchain deployment in the industry. The findings highlight blockchain's transformative potential in enterprise operations and financial systems.

By providing actionable insights and recommendations, this thesis underscores how Hyperledger Fabric could reshape existing banking structures and addresses the feasibility of transitioning from traditional microservices to innovative blockchain models.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
MVC	Model-View-Controller
SOAP	Simple Object Access Protocol
SWIFT	Society for Worldwide Interbank Financial Telecommunication
RTGS	Real Time Gross Settlement
AWS	Amazon Web Service
MSP	Membership Service Provider

TABLE OF CONTENTS

CHAPTER 1 :

INTRODUCTION

1.1 Background

1.1.1 Current Banking Infrastructure

Contemporary banking infrastructure is predominantly characterized by traditional service-oriented architectures (SOA), which have progressively transitioned into microservice-based architectures to achieve increased scalability and modularity. Microservices decompose applications into discrete, autonomous services capable of communicating through lightweight APIs. This methodology is extensively adopted owing to its adaptability, scalability, and competence in integrating with legacy systems while permitting real-time transaction processing.Notably, within the financial services sector, streaming platforms are increasingly employed to manage high-frequency data processing, thereby enhancing transaction throughput and fraud detection (Mallidi, Sharma, and Vangala, 2022). Nevertheless, despite these advancements, this paradigm encounters specific challenges, such as dependency management, inter-service communication, and the substantial costs associated with maintaining multiple service modules.

1.1.2 Challenges in Current Architectures

The intrinsic challenges associated with conventional architectures encompass the maintenance of transaction consistency across distributed systems and the assurance of data privacy. While Service-Oriented Architecture (SOA) and microservices enhance operational efficiency, their dependency on centralized databases renders them susceptible to becoming single points of failure or bottlenecks. Moreover, the imperative for real-time processing has revealed limitations in scaling these systems effectively for large-scale banking applications. Research has identified the potential of emerging paradigms, such as blockchain, to address some of these fundamental issues by providing decentralized transaction management and improved data integrity (Konkin and Zapechnikov, 2021).

These challenges underscore the need to investigate alternative architectural solutions to improve banking infrastructures.

1.1.3 Introduction to Blockchain Technology

Blockchain technology is increasingly recognized as a transformative innovation for the management of distributed systems. Its decentralized and immutable ledger offers distinctive advantages, particularly in the realm of finance. A significant advancement in this domain is the Hyperledger Fabric framework, which is specifically designed for enterprise applications, facilitating permissioned blockchains that ensure transaction confidentiality while ensuring transparency. Such frameworks support secure multiparty collaborations and are capable of addressing scalability and privacy challenges present in conventional banking systems (Quan, Wahab, Fadila, Aun, S. K. Luk, et al., 2024). Furthermore, blockchain technology incorporates sophisticated cryptographic methods, such as zero-knowledge proofs, to safeguard transactional confidentiality, making it suitable for handling sensitive financial transactions (Konkin and Zapechnikov, 2021).

1.1.4 The Convergence of Blockchain and Banking

The integration of blockchain technology into banking systems has the potential to revolutionize conventional infrastructure. Smart contracts, which enable the automation of processes based on predefined parameters, can substantially enhance operational efficiency in areas such as supply chain financing and payment settlements(Johnson, Dandapani, and Sharokhi, 2024). In addition, private blockchain networks, as exemplified by those developed on Ethereum and Hyperledger Fabric, allow financial institutions to maintain control while benefiting from the advantages of decentralization. Recent scholarly research highlights successful implementations of blockchain frameworks in sectors analogous to banking, demonstrating their viability and scalability (Quan, Wahab, Fadila, Aun, S. K. Luk, et al., 2024). This amalgamation of traditional and blockchain-based architectures signals a promising path for the advancement of financial infrastructure.

1.2 Problem Statement

1.2.1 The Problem

Traditional banking systems are predominantly dependent on centralized architectures, including MVC frameworks and SOAP APIs. Although these systems effectively facilitate routine transactions, they face substantial challenges, such as scalability limitations, elevated transaction costs, and susceptibility to fraudulent activities. As financial operations increasingly demand rapid, secure, and transparent systems, the inadequacies of these traditional architectures have become more pronounced. The centralized nature of such systems often leads to bottlenecks, higher maintenance expenses, and inefficiencies that hinder their ability to meet the evolving demands of the financial sector (Mallidi, Sharma, and Vangala, 2022). Blockchain technology has emerged as a viable alternative to traditional systems, offering decentralized transaction management along with features like improved security and transparency. This technology presents the potential to transform banking infrastructure. Frameworks such as Hyperledger Fabric cater specifically for enterprise use, enabling permissioned blockchains with strong privacy controls. However, transitioning from conventional centralized systems to blockchain-based solutions presents challenges, including complexities in integration, scalability issues, and regulatory compliance concerns (Quan, Wahab, Fadila, Aun, S. K. Luk, et al., 2024). This underscores the need for a thorough evaluation of the suitability of blockchains for contemporary banking.

1.2.2 Knowledge Gaps

1. **Comparative Analysis of Architectures:** Recent studies underscore the advantages of blockchain technology; however, they lack comprehensive comparative analyses between traditional microservices-based architectures and blockchain frameworks such as Hyperledger Fabric. Essential factors, including transaction speed, cost-effectiveness, and security, remain insufficiently explored within real-world banking scenarios (Konkin and Zapech-

nikov, 2021).

2. **Operational Feasibility:** Research frequently neglects the technical and organizational challenges associated with integrating blockchain into existing banking infrastructures. There are no thorough investigations into the architectural and operational difficulties encountered during these transitions (Quan et al., 2024).

3. **Regulatory Challenges:** While blockchain's capacity to enhance transparency is extensively documented, there exists a scarcity of research concerning the governance and regulatory compliance issues within the banking industry. This deficiency impedes banks from comprehending how to align blockchain adoption with stringent financial regulations (Konkin and Zapechnikov, 2021).

4. **Scalability in High-Volume Banking:** Although blockchain's scalability challenges are recognized, there is a limited focus on its performance under the duress of high-transaction environments characteristic of major banking institutions and payment systems (Mallidi, Sharma, and Vangala, 2022).

5. **Long-Term Sustainability:** Research often emphasizes the immediate benefits of blockchain, yet seldom addresses its long-term sustainability within banking, specifically regarding energy consumption, infrastructure maintenance, and adaptation to evolving technological standards (Quan, Wahab, Fadila, Aun, S. K. Luk, et al., 2024).

1.2.3 The Statement

This investigation aims to determine whether Hyperledger Fabric is ready for enterprise adoption within the banking sector and if it possesses the potential to supplant current microservices-based architectures. The study endeavors to assess Hyperledger Fabric's performance concerning critical parameters such as transaction efficiency, scalability, and security in comparison to conventional banking systems. By addressing extant knowledge deficits, this research intends to furnish insights into the practical applicability of blockchain in authentic banking contexts. Through a combined approach of comparative analysis and practical implementation, the study will investigate whether blockchain frameworks can overcome integration and regulatory challenges while maintaining scalability and long-term viability. The findings will elucidate whether Hyperledger Fabric constitutes a feasible alternative to traditional architectures and will delineate the requisite steps for successful deployment in the banking sector. This methodology endeavors to bridge the existing comprehension gap and aid the transition towards blockchain-based banking systems.

1.3 Aim And Objective

1.3.1 Aim

The aim of this study is to explore the transformative potential of blockchain technology in modern banking systems. It seeks to investigate how blockchain, as a decentralized ledger framework, can address inefficiencies and security vulnerabilities inherent in traditional banking infrastructures. By focusing on the ability of blockchain to improve transaction transparency, reliability, and efficiency, this study aims to provide a complete understanding of its readiness for enterprise adoption in the financial sector.

1.3.2 Objectives

This research will compare blockchain-based systems, specifically Hyperledger Fabric, with conventional microservices architectures in banking. The evaluation will focus on key metrics such as transaction speed, operational cost, and fraud prevention. Through a detailed analysis of the strengths and weaknesses of both systems, the study aims to determine the applicability of blockchain in real-world banking scenarios. Furthermore, the research will evaluate the scalability, regulatory compliance, and long-term sustainability of blockchain solutions, providing actionable insights into their viability as replacements for existing architectures.

1.4 Research Questions

To address the specific research objectives, the following research questions are proposed:

1. *Is blockchain technology in its current state a viable alternative for the modern banking ecosystem?*
2. *In what areas is blockchain technology lacking when it comes to mainstream adoptions?*
3. *What steps can we take to mitigate the disadvantages of this technology and prepare for its mainstream adoption?*

1.5 Scope

The scope of this research is broad, encompassing not only the operational differences between blockchain and traditional systems, but also the broader economic and regulatory implications. By examining transaction speed, cost efficiency, security protocols, and compliance challenges, this research addresses key areas where blockchain can have a competitive advantage over legacy systems(Idris et al., 2023). In addition, it explores the integration of blockchain technology into existing infrastructures, particularly in domains such as cross-border payments, fraud detection, and digital asset management.

1.6 Significance

The significance of comparing blockchain and traditional banking transaction systems is important in the current financial landscape. As blockchain technology continues to evolve and gain traction in various sectors, the banking industry is at a critical juncture. The evaluation of the potential of blockchain to improve efficiency, security and operational processes is of paramount importance, as financial institutions are under increasing pressure to modernize systems, reduce transaction costs, and improve transparency. This research is particularly relevant as customers demand faster and more secure services, and the banking sector is faced with the challenge of meeting these expectations(Tressa and Priya, 2023).

1.7 Structure

This study is structured to systematically compare the service-based microservices architecture and the hyperledger fabric-based blockchain architecture for banking systems, focusing on critical metrics relevant to financial operations. The research begins with the development of two prototype architectures: a traditional microservices framework designed around modular service components and a blockchain framework that takes advantage of the Hyperledger Fabric platform. Both architectures will be designed to simulate real-world banking scenarios, such as interbank transactions, customer account management, and payment processing.(Zhang et al., 2020)
The study evaluates both prototypes in several key metrics. **Transaction speed** will be measured by benchmarking the time required to process a given volume of transactions under identical conditions. Similarly, **the time taken to settle interbank transactions** will be compared to assess the efficiency of transaction finality. **Security** will be evaluated by examining the robustness of data encryption in transit and the protection mechanisms for stored data, such as access controls and tamper resistance (Konkin and Zapechnikov, 2021). The **complexity of the setup** will be evaluated based on the effort required to configure and deploy each architecture, while the **ease of migration from monolithic systems** will be measured by analyzing the compatibility of both architectures with legacy banking systems (Quan, Wahab, Fadila, Aun, S. K. L. Luk, et al., 2024). Other metrics, such as scalability, operational cost, and long-term sustainability, will also be examined to provide a holistic comparison. For example, streaming platform architectures in banking have been explored for scalability, offering insights into how these architectures manage high transaction volumes (Mallidi et al., 2022). This study will use these insights to contextualize the scalability of blockchain systems. By examining these factors, the research aims to highlight the trade-offs between traditional and blockchain-based banking infrastructures,

providing actionable insights for their adoption in enterprise banking environments.

CHAPTER 2 :

LITERATURE REVIEW

2.1 Technology in Banking

The evolution of technology has been an essential component in the transformation of banking systems worldwide. Before the advent of computer systems, banks relied on manual processes for customer management, bookkeeping, and transaction processing. These processes were labor intensive and prone to errors, limiting the scalability and efficiency of financial institutions. However, with technological advancements, banks gradually adopted computerized systems, which streamlined operations, improved accuracy, and enhanced security. Today, technology continues to redefine banking, shifting from centralized, service-based architectures to innovative solutions like blockchain that promise decentralized, secure, and transparent operations. This section explores the key concepts of traditional banking, the transition to digital systems, and the technological infrastructure that underpins modern banking.

2.1.1 Concepts of banking

At its core, banking revolves around fundamental operations such as customer management, cash handling, bookkeeping, deposits, withdrawals, and ensuring security.

1. **Customer Management**: Banks have historically prioritized customer relationships, maintaining manual ledgers to record account details, deposits, and withdrawals. Early systems relied on physical documents that required substantial human effort to ensure accuracy. With digitalization, customer data management was shifted to centralized databases, which streamlined account opening, balance inquiries, and service requests. Digital tools like CRM systems improved customer experiences, paving the way for personalized banking solutions. However, centralized customer data has raised concerns about data breaches and security vulnerabilities, highlighting the

importance of advancements in cryptographic technologies(Muneeb et al., 2022).

2. **Cash Handling**: Cash management in traditional banking included manual counting, recordkeeping, and security measures. The introduction of Automated Teller Machines (ATMs) in the late twentieth century revolutionized cash handling, enabling customers to access funds anytime. These systems were further integrated with centralized banking platforms, allowing real-time updates to customer accounts. Blockchain-based banking now explores secure and traceable cash handling alternatives, potentially reducing operational costs and fraud risks.

3. **Bookkeeping and Security**: Bookkeeping was a tedious process in early banking, relying on physical ledgers and manual reconciliations. Digital systems transformed bookkeeping by automating these processes, ensuring accuracy and reducing human error. Security, initially reliant on physical safeguards, evolved into sophisticated cybersecurity frameworks to protect digital assets. Blockchain and encryption technologies are now being investigated for secure record keeping, offering tamper-resistant ledgers that enhance transparency(ibid.).

2.1.2 Ledger

1. **The Concept of a Ledger in Banking**: A ledger is the foundation of banking and serves as a comprehensive record of all financial transactions. In traditional banking, the physical ledgers maintained by the clerks documented every debit and credit, forming the backbone of financial accountability. However, the manual nature of these systems was prone to errors, inefficiencies, and fraud.

2. **Transition to Digital Systems**: With the advent of computerized systems, physical ledgers were replaced by digital ones, enabling banks to manage accounts, transactions, and reports more effectively. Centralized databases allowed real-time updates, reducing the time required for reconciliation. Despite these advancements, centralized systems introduced risks such as data

breaches and single points of failure. Emerging technologies like blockchain promise to address these challenges by providing decentralized, tamper-resistant ledgers. Hyperledger Fabric, for instance, offers an enterprise-grade framework that enables permissioned ledger management, maintaining both transparency and privacy(Quan, Wahab, Fadila, Aun, S. K. L. Luk, et al., 2024)

2.1.3 Transactions

1. **Intra-Bank Transactions**: Transactions within the same bank are relatively straightforward, involving the transfer of funds between accounts under a single ledger. Early systems required manual entries for each transfer, which were later reconciled at the end of the day. Modern banking systems automate these processes, enabling instant fund transfers through integrated digital platforms. However, centralized transaction management systems remain vulnerable to operational delays during peak loads.

2. **Inter-Bank Transactions**: Interbank transactions involve the transfer of funds between different financial institutions and are inherently more complex. Traditionally, these transactions were managed through manual clearinghouses, which required significant time and effort. The introduction of electronic fund transfer (EFT) systems, such as SWIFT and RTGS, revolutionized interbank payments, reducing processing times, and improving accuracy. Despite these advancements, interbank transactions still face inefficiencies due to intermediary dependencies. Blockchain-based solutions, such as those proposed by Konkin and Zapechnikov, 2021, aim to eliminate intermediaries, allowing faster settlement times and reduced costs.

3. **Fraud Prevention and Data Security**: As transaction volumes increased, fraud prevention became a critical concern. Traditional systems implemented security measures such as access controls and transaction logs. With digitalization, banks adopted encryption, firewalls, and fraud detection algorithms to protect customer data and prevent unauthorized transactions. Blockchain technology improves fraud prevention by using cryptographic methods to secure transaction data, ensuring transparency and immutability across the

network. Studies highlight the potential of blockchain to reduce fraud in interbank transactions by creating tamper-proof records(Muneeb et al., 2022).

2.1.4 Conclusion

Technology has been instrumental in transforming banking from labor-intensive manual operations to sophisticated digital ecosystems. From customer management and bookkeeping to inter-bank transactions, each facet of banking has benefited from technological advancements. However, centralized systems still face challenges in scalability, security, and efficiency. Blockchain technology, particularly frameworks like Hyperledger Fabric, offers promising alternatives to traditional architectures. By addressing current limitations, blockchain has the potential to redefine the future of banking, ensuring secure, efficient, and transparent financial operations.

2.2 Legacy Systems

For decades, legacy systems have been the foundational infrastructure of banking operations, facilitating financial institutions in the execution of transactions, maintenance of customer records, and ensuring compliance with regulatory requirements. Traditionally, these systems have been constructed on robust mainframe infrastructures, celebrated for their exceptional reliability, security, and computational capacity. However, as the requirements for scalability, flexibility, and real-time processing have intensified, the limitations inherent in mainframe systems have become increasingly apparent. With the progression of software development methodologies, legacy systems have undergone an evolution incorporating monolithic architectures, Java-based frameworks, and standardized design patterns. This section examines the principal characteristics of these systems, their developmental trajectory, and the challenges they face in contemporary banking requirements.

2.2.1 Monolithic Systems

Monolithic architecture refers to a software design paradigm in which all components of an application, user interface, business logic, and database management, are tightly integrated into a single unit. This approach was prevalent in early banking systems because of its simplicity and the centralized nature of operations. Banks used these systems to process transactions, maintain ledgers, and manage customer accounts under a unified framework.

Despite its initial efficiency, the monolithic architecture struggled to adapt to changing business needs. The tightly coupled nature of these systems often led to operational bottlenecks, especially during high-demand periods. For example, deploying a minor update required re-deploying the entire system, resulting in downtime and resource inefficiencies (Navarro, 2022). Moreover, as the volume of transactions and the complexity of services increased, monolithic systems faced challenges in scaling horizontally, highlighting the need for more modular and distributed architectures.

2.2.2 Java and Jakarta EE Architectures with JSPs and EJBs

The adoption of Java Enterprise Edition (now Jakarta EE) revolutionized the development of banking applications. JavaServer Pages (JSPs) allowed dynamic content generation, facilitating the creation of interactive banking portals. These portals enabled customers to perform transactions, view account details, and manage funds through user-friendly interfaces.

Enterprise JavaBeans (EJBs), another core component of Jakarta EE, introduced a modular approach to managing business logic. In banking, EJBs were leveraged for critical functionalities such as fund transfers, loan approvals, and risk assessments. By decoupling business logic from the user interface, EJBs enhanced maintainability and scalability. However, the implementation complexity of JSPs and EJBs, coupled with the monolithic architecture's limitations, restricted their ability to meet modern banking requirements(Muslimin, Fajar, and

Meyliana, 2020).

2.2.3 Disadvantages of Monolithic Systems

Although monolithic systems served as a cornerstone for early banking, they posed several challenges in adapting to modern technological demands:

1. **Scalability Issues:** The tightly coupled components of monolithic systems make it difficult to scale specific functionalities independently. As a result, banks must allocate additional resources to the entire system, even for minor scaling needs(Patil, Pramod, and S, 2022).

2. **Maintenance and Deployment:** Updating a monolithic application is cumbersome, as any change requires the redeployment of the entire system. This process often leads to increased downtime and operational risks.

3. **Integration Challenges:** Integrating new services or third-party applications with monolithic systems is challenging due to their rigid architecture. This limitation affects the adoption of innovative technologies like blockchain and real-time analytics(Attia and Abed, 2024).

4. **High Resource Utilization:** The centralized nature of monolithic systems demands substantial computational resources, leading to higher operational costs.

2.2.4 Design Patterns in Monolithic Systems

Despite their limitations, monolithic systems were often built using standardized design patterns to ensure efficiency, maintainability, and scalability. These patterns are commonly represented through Unified Modeling Language (UML) diagrams:

1. **Singleton Pattern:** This pattern ensured a single instance of a resource, such as a database connection, was shared across the application as shown in figure 2.1. This approach reduced redundancy and optimized resource utilization.

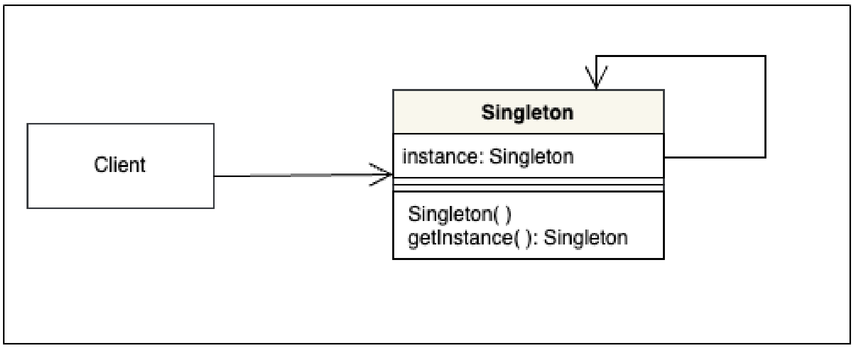


Figure 2.1: Singleton Pattern

2. **Layered Architecture Pattern:** Monolithic systems were organized into layers as seen in figure 2.2, separating the user interface, business logic, and data access functionalities. This division improved code organization and debugging.

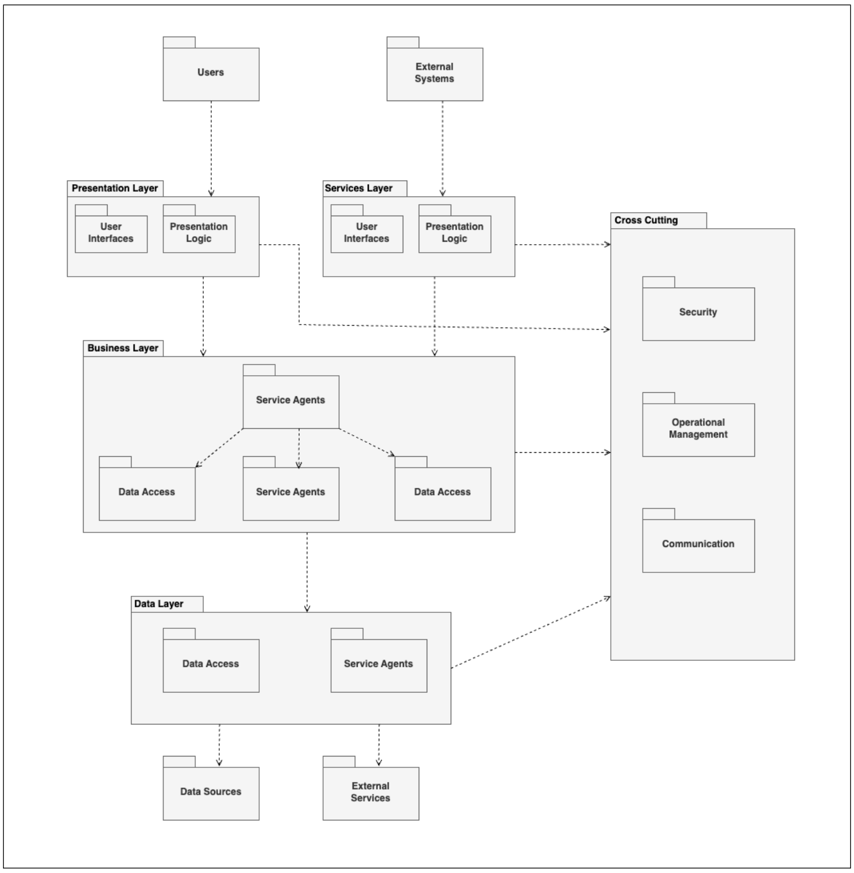


Figure 2.2: Layered Architecture Pattern

3. **Factory Method Pattern:** This pattern dynamically created objects based on specific requirements, such as generating customer profiles or transaction logs as seen in figure 2.3.

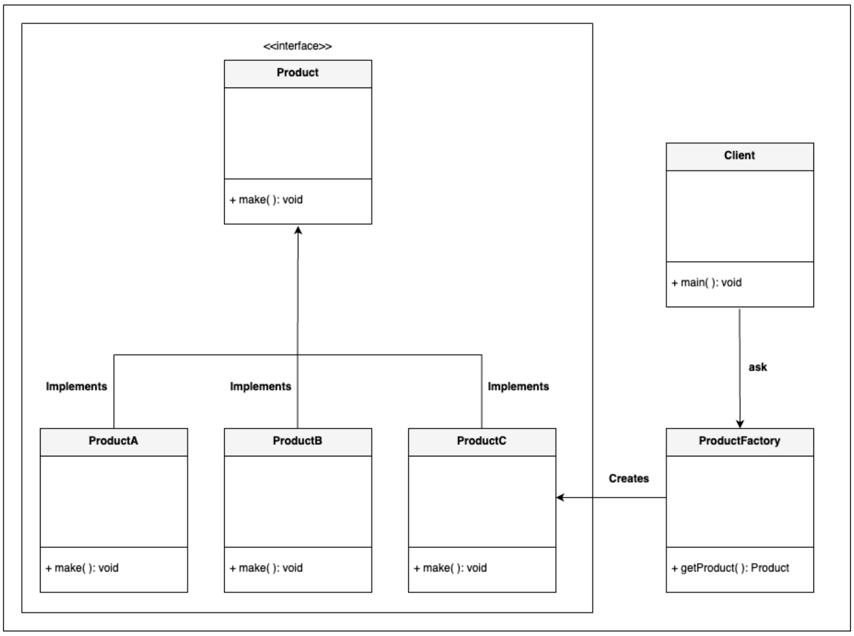


Figure 2.3: Factory Pattern

4. **Observer Pattern:** Real-time updates across system modules were managed using the Observer pattern, ensuring data consistency as seen in figure 2.4. For instance, updates in account balances reflected instantly in all relevant modules.

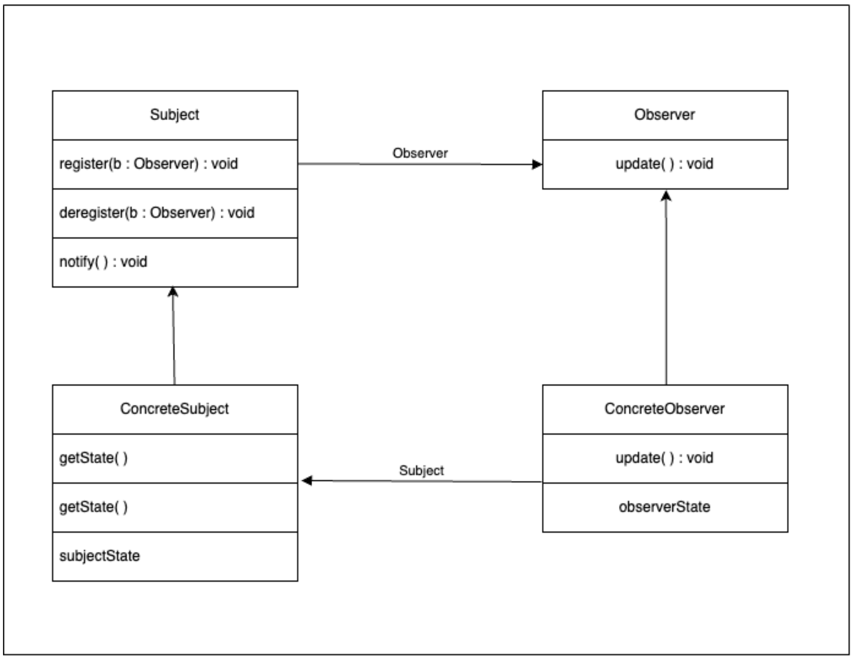


Figure 2.4: Observer Pattern

Legacy banking systems, established on monolithic architectures and Java-based frameworks, were instrumental in forming the technological cornerstone of the financial sector. Nonetheless, these systems exhibit intrinsic limitations in terms of scalability, deployment, and integration, thereby necessitating a shift towards more flexible and modular solutions. While the implementation of design patterns has imparted a measure of maintainability and efficiency, it remains insufficient in fully addressing the exigencies of contemporary banking requirements. This progression underscores the imperative for distributed architectures, such as micro-services and blockchain-based solutions, which provide the scalability, resilience, and adaptability that are imperative for modern financial operations.

2.3 Migration to Micro-services

The transition from legacy systems to micro-service-based architectures has significantly transformed the banking sector. Micro-services represent a contemporary software design paradigm that structures applications as an assemblage of loosely coupled, independently deployable services. In contrast to monolithic systems, micro-services facilitate scalability, resilience, and accelerated development cycles, making them an appealing option for banks seeking to fulfill the demands of digital transformation. As financial institutions strive to modernize their infrastructure frameworks, migration to micro-services has become indispensable to address the growing demands for real-time data processing, enhanced customer experiences, and seamless integration with third-party services.

2.3.1 Concept of micro-services

Micro-services are an architectural style that structures an application as a collection of small, autonomous services, each responsible for a specific business function. This design promotes modularity, enabling teams to develop, deploy, and scale individual services independently. For example, in banking, services for account management, transaction processing, and fraud detection can be developed and updated without affecting other components.

The hallmark of micro-services is their independence, achieved through the use of lightweight communication protocols like REST APIs or messaging queues. Each service is deployed in its container, allowing it to operate independently of the underlying infrastructure. This approach contrasts sharply with monolithic systems, where all functions are interdependent, making updates and scaling cumbersome(Bucchiarone et al., 2018).

Key benefits of micro-services in banking include:

- **Scalability:** Services experiencing high demand can be scaled independently, optimizing resource allocation.
- **Resilience:** Failures in one service do not affect the entire application, ensuring continuity.
- **Agility:** Faster development and deployment cycles enable banks to innovate and respond to market changes quickly.

2.3.2 Frameworks Used In Micro-servies

The adoption of micro-services has been facilitated by a variety of frameworks and tools that provide robust environments for development and deployment. Some widely used frameworks in the banking industry include:

1. **Spring Boot:** Spring Boot simplifies the development of micro-services by offering pre-configured templates and tools for building scalable and production-ready applications. Its ability to integrate seamlessly with cloud platforms makes it a popular choice for banks transitioning from legacy systems(Larsson, 2021).

2. **Docker and Kubernetes:** Docker enables containerization of micro-services, ensuring consistent deployment across different environments. Kubernetes complements Docker by managing container orchestration, automating tasks like scaling, load balancing, and self-healing. Together, these tools ensure high availability and scalability.

3. **Apache Kafka:** Kafka is a distributed messaging system used for real-time data streaming. In banking, Kafka facilitates inter-service communication

by enabling asynchronous messaging, ensuring seamless data exchange between micro-services.

4. **Quarkus:** Quarkus is a Java framework designed for cloud-native micro-services. It provides faster startup times and lower memory usage, making it ideal for resource-constrained environments in banking.

2.3.3 Migration to micro-services

Migrating from legacy systems to micro-services is a complex yet rewarding process that involves re-architecting existing systems to adopt a service-oriented approach. The migration process typically includes the following steps:

1. **Assessment and Planning:** Banks must evaluate their existing systems to identify components that can be modularized. Key considerations include the complexity of current applications, dependencies, and potential challenges in data migration.

2. **Gradual Transition:** A phased approach is often recommended, where legacy systems coexist with micro-services during the migration. This strategy minimizes disruptions and allows for iterative testing and optimization(Capuano and Muccini, 2022)

3. **Data Decoupling:** One of the most critical steps in migration is decoupling data from monolithic systems. Databases are often broken into smaller, service-specific databases, ensuring each micro-service has its data repository.

4. **Testing and Deployment:** Automated testing frameworks are crucial during migration to ensure that each micro-service functions as expected. Continuous integration and continuous deployment (CI/CD) pipelines facilitate faster delivery and minimize errors.

2.3.4 Design Patterns in Micro-services

Micro-services architecture is supported by several design patterns that enhance scalability, fault tolerance, and maintainability.

1. **API Gateway Pattern:** The API Gateway acts as a single entry point for all client requests, routing them to the appropriate micro-services. This pattern simplifies client interactions and enables features like authentication, load balancing, and caching.
2. **Database per Service Pattern:** Each micro-service maintains its database, ensuring data encapsulation and independence. This pattern eliminates the need for shared databases, reducing inter-service dependencies.
3. **Event-Driven Architecture:** Micro-services communicate asynchronously using events, enabling real-time data exchange and reducing coupling. Kafka or RabbitMQ is often used for implementing this pattern.
4. **Service Discovery Pattern:** Service discovery enables micro-services to locate each other dynamically, ensuring seamless communication in distributed environments. Tools like Eureka or Consul are commonly used for service discovery.
5. **Circuit Breaker Pattern:** To enhance resilience, the Circuit Breaker pattern prevents cascading failures by temporarily stopping requests to a service when it detects a fault.

The transition from legacy systems to micro-services represents a paradigm shift in banking infrastructure, enabling financial institutions to meet the demands of a digital-first economy. Micro-services offer unparalleled flexibility, scalability, and resilience, addressing the limitations of monolithic systems. Frameworks like Spring Boot, Kubernetes, and Kafka provide robust environments for developing and managing micro-services, while design patterns such as API Gateway and Event-Driven Architecture ensure their efficiency and maintainability. Although the migration process is complex, its benefits far outweigh the challenges, making micro-services a cornerstone of modern banking operations.

2.4 Blockchain Technology

2.4.1 Emergence of Distributed Systems

Decentralized systems have fundamentally transformed the digital landscape by contesting traditional centralized operational models. These systems allocate control and decision making across a network, thereby obviating dependence on a solitary governing entity. Centralized databases in conventional systems are susceptible to risks, including single point of failure, data breaches, and scalability inefficiencies. Decentralized systems effectively mitigate these challenges by using distributed networks in which nodes collectively manage and validate data. This transition has profoundly influenced various industries, notably finance, providing transparency, resilience, and operational efficiency.

The introduction of blockchain technology has accelerated the adoption of decentralized systems, enabling applications such as digital currencies, secure data management, and efficient transaction processing. Unlike centralized systems that depend on trust in a singular authority, decentralized systems distribute trust across a network. This reduces vulnerabilities and creates systems resistant to fraud and tampering (Fedorov et al., 2021). Decentralized networks, empowered by consensus algorithms, ensure data integrity without requiring a central authority, paving the way for innovations such as smart contracts and tokenized economies.

2.4.2 Idea of Decentralized or Zero Trust Systems

Zero trust systems extend the principles of decentralization by advocating a security model where trust is not inherent but must be continually verified. These systems are built on the premise that no entity, internal or external, should be trusted by default. Instead, access is granted based on continuous authentication, authorization, and validation of credentials.

Blockchain technology exemplifies the zero-trust paradigm through its reliance on cryptographic proofs and consensus mechanisms. Every transaction on a blockchain network is verified by participating nodes, ensuring trust is derived from the system's integrity rather than external authorities. For example, private blockchain systems employing zero-trust architectures ensure secure and tamper-resistant data exchange by enforcing strict access controls and real-time validation (Chang et al., 2024).This approach is particularly valuable in sectors like banking, where the confidentiality and integrity of transactions are paramount

Zero trust principles also enhance resilience against insider threats and external attacks. By adopting these architectures, blockchain networks align with the growing demand for robust, scalable, and secure digital ecosystems.

2.4.3 Concept of Blockchain

Blockchain is a decentralized and immutable ledger that records transactions across multiple nodes in a network. The innovation lies in its structure: data is stored in blocks that are linked chronologically to form a chain. Each block contains a timestamp, transaction data, and a cryptographic hash of the previous block, ensuring data integrity and resistance to tampering(Bashir, 2023).

Initially conceived for digital currencies such as Bitcoin, blockchain technology has expanded its applicability to encompass domains like supply chain management, identity verification, and financial transactions. Its decentralized architecture ensures that control is not vested in a single entity, thereby promoting transparency and safeguarding security (Wan, Liu, and Xiao, 2020). Blockchain functions through consensus mechanisms, including proof-of-work (PoW) or proof-of-stake (PoS), to validate transactions and prevent incidents of double-spending.

In the context of banking, blockchain provides a secure alternative to conventional centralized systems by facilitating the real-time processing and verification of transactions. This diminishes dependency on intermediaries, reduces transaction costs, and enhances operational efficiency. Additionally, the transparency and immutability inherent in blockchain furnish audit trails, thereby augmenting regulatory compliance within financial operations.

2.4.4 Permissioned Ledger

Permissioned ledgers, a subset of blockchain, are designed for enterprise use, offering controlled access to participants. Unlike public blockchains, where anyone can join and validate transactions, permissioned ledgers restrict access to authorized entities. This structure is particularly advantageous in industries requiring confidentiality, such as banking and healthcare.

Hyperledger Fabric exemplifies a leading framework for permissioned ledgers, facilitating customizable governance models that safeguard data privacy while simultaneously maintaining transparency among sanctioned participants. Within permissioned networks, consensus mechanisms are calibrated for efficiency rather than decentralization, thereby rendering them appropriate for environments necessitating high throughput (Vidwans et al., 2022). By integrating features such as access controls, encrypted communication channels, and a modular architecture, permissioned ledgers effectively harmonize the imperatives of privacy and collaboration. These characteristics render them particularly suitable for applications such as interbank settlements, secure data sharing, and adherence to regulatory frameworks. Studies underscore their potential to mitigate transaction delays and operational costs within financial systems while ensuring robust security measures.

2.4.5 Smart Contracts

Smart contracts are self-executing contracts with terms encoded directly into code. These contracts automate processes, ensuring that pre-defined conditions are met before actions are executed. Running on blockchain networks, smart contracts eliminate the need for intermediaries, reducing costs and potential errors in execution(Zand, Wu, and Morris, 2021).

In the banking sector, smart contracts possess significant transformative potential in domains such as loan disbursements, trade finance, and payment settlements. For instance, a smart contract is capable of autonomously releasing funds when the stipulated conditions in a trade agreement are satisfied, thereby optimizing processes and minimizing disputes (Singh et al., 2023). The capacity to generate immutable and transparent contracts bolsters trust among stakeholders

. Nonetheless, the complexity associated with the design and deployment of smart contracts necessitates the implementation of robust frameworks for development and testing. Vulnerabilities in the code have the potential to be exploited, underscoring the critical importance of rigorous security measures. Platforms such as Hyperledger Fabric offer tools for the creation and management of secure smart contracts, facilitating their integration into enterprise applications.

2.5 Hyper Ledger Project

Hyperledger Fabric constitutes a robust framework designed to facilitate the development of enterprise blockchain solutions. It offers a modular architecture that enables organizations to tailor their blockchain networks to align with particular use cases, with an emphasis on scalability, privacy, and flexibility. This section explores the fundamental concepts and components of Hyperledger Fabric, offering an in-depth understanding of its operational structure and practical utility.

2.5.1 Key Concepts

Membership services

The Membership Services in Hyperledger Fabric are responsible for the administration of identities and permissions of participants within the network. Contrasting with public blockchains, where any node may freely participate, Fabric adopts a permissioned model that restricts access exclusively to authenticated entities. The Membership Service Provider (MSP) supervises the identity management procedure, ensuring adherence to the governance protocols of the network. This regulated access enhances privacy, which is crucial for sectors such as banking, where compliance with regulatory standards is imperative(Attia and Abed, 2024).

Blockchain services

Fabric's blockchain services ensure data immutability and transparency. Transactions are recorded as blocks and appended to the ledger in a sequential order. Unlike traditional blockchain models that rely on a proof-of-work consensus, Fabric uses more efficient protocols like Practical Byzantine Fault Tolerance (PBFT). This approach significantly reduces transaction latency and energy consumption, making it suitable for enterprise applications (Kuzlu et al., 2019).

Smart Contract services

In Hyperledger Fabric, smart contracts, known as chaincode, facilitate the automation of business logic by executing predefined conditions, thereby ensuring trust and removing intermediaries. Chaincode is executed within a Docker container, which provides a secure and isolated environment. Its applications encompass automation of banking transactions, optimization of trade finance, and improvement of regulatory compliance through audit trails (Shah et al., 2023).

Fabric offers a comprehensive suite of Application Programming Interfaces (APIs) and Command Line Interfaces (CLIs) for network interaction. APIs enable developers to create and deploy decentralized applications (DApps), whereas CLIs provide tools for network configuration, transaction submission, and ledger examination. This dual interface system simplifies network management and improves accessibility for developers and administrators (Padmavathi et al., 2024).

2.5.2 Components

Peers/nodes

Peers are the backbone of a Fabric network, responsible for validating transactions, maintaining the ledger, and hosting smart contracts. Each peer has a unique identity managed by the MSP. Peers are categorized into endorsing peers, which validate transactions, and committing peers, which update the ledger. This division of labor ensures network efficiency and scalability (Dimou, Choumas, and Korakis, 2023).

Clients

Clients interact with the blockchain network by submitting transaction proposals. These proposals are sent to endorsing peers for validation before being added to the ledger. Client applications can be customized to suit specific business needs, enabling seamless integration with existing systems (Kaushal and Kumar, 2023).

Channels

Channels enable private communication between subsets of network participants. This feature is crucial for scenarios where sensitive data must be shared selectively, such as interbank settlements. Channels ensure data confidentiality while maintaining the integrity of the overarching network (Vidwans et al., 2022).

World State Database

The World State Database is responsible for storing the ledger's current state, thereby providing a rapid means for querying data. Fabric accommodates a variety of database options, including CouchDB for handling complex queries and LevelDB for more lightweight implementations. This adaptability permits businesses to optimize their networks according to operational demands (Laishevskiy, Barger, and Gorgadze, 2023).

Private Data Collections

Private Data Collections significantly enhance data privacy by storing sensitive information outside the main blockchain (off-chain). Access to these collections is restricted to authorized participants, thereby ensuring confidentiality while upholding the network's integrity. This functionality is particularly pertinent in banking applications, where data protection is of critical importance (Taha and Alanezi, 2023).

Transactions

Transactions within Fabric adhere to a tripartite process comprising proposal, endorsement, and commitment phases. Transactions are initiated through proposals by clients, subsequently validated by endorsing peers, and ultimately committed to the ledger. This methodology guarantees consensus and precludes unauthorized updates, thereby reinforcing trust among network participants (Zhao, 2022).

Membership Service Provider (MSP)

The MSP manages participant identities through digital certificates. These certificates are issued by a Certificate Authority (CA) and validate user credentials. The MSP framework ensures secure and reliable identity management, which is essential for maintaining network integrity (Shah et al., 2023).

Smart Contracts

Smart contracts in Fabric automate processes by embedding business rules within chaincode. These contracts are executed on endorsing peers, ensuring compliance with predefined conditions. Use cases include automating loan approvals and facilitating cross-border payments (Mathwale, 2023).

Crypto Service Provider

The Crypto Service Provider (CSP) underpins the security architecture of Fabric, offering cryptographic functions such as encryption, digital signatures, and hashing. These services ensure data confidentiality, authenticity, and integrity, aligning with the stringent security requirements of enterprise applications (Sun and Yuan, 2022).

CHAPTER 3 :

RESEARCH METHODOLOGY

3.1 Introduction

The objective of this research is to compare the performance, scalability, and security of two distinct banking architectures:

- **Service-Based Microservices Architecture** using Spring Boot and MySQL.
- **Permissioned Blockchain Architecture** using Hyperledger Fabric.

This comparison will provide insights into which architecture is more suitable for modern banking applications under specific conditions.

3.2 Objectives

1. **Performance Evaluation:** Measure and compare the response time, throughput, and resource utilization.
2. **Scalability Analysis:** Assess how each architecture handles increasing loads.
3. **Security Assessment:** Evaluate the security features inherent to each architecture.
4. **Resource Utilization:** Monitor CPU and memory usage during load testing.

3.3 Experimental Setup

3.3.1 Hardware Configuration

A Workstation with the following configuration:

- AMD Ryzen 9 9950X CPU (16 Cores 32 Threads)
- 64 GB DDR5 RAM
- Operating System: Ubuntu 24.01

- NVIDIA RTX 2060 6GB Graphics

3.3.2 Software Tools

- **Spring Boot 3** for microservices implementation
- **MySQL** as the relational database servie.
- **HyperLedger fabric** for blockchain implementation.
- **Jmeter** for load testing.

3.3.3 Constraints

As we have no credits for cloud services like Azure or AWS we will limit our testing on our local machine/workstation.

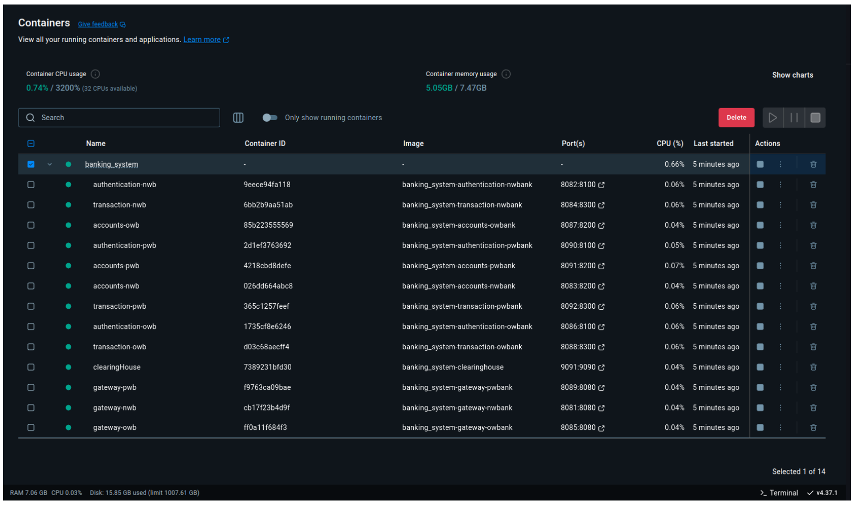
3.4 Data Preparation

- **Sample Data:** Utilize a custom-generated dataset resembling real customer data (as provided).
- **Data Scaling:** Expand the sample dataset to simulate a realistic banking environment.
- **Data Storage:**
  - For MySQL, import the dataset into relational tables.
  - For Hyperledger Fabric, encode the data into ledger entries via chaincode (smart contracts).

3.5 Implementation

For this study, two systems were developed and tested:

- **Banking System**: A microservices-based architecture implemented using Spring Boot and MySQL, available on GitHub:Banking System.
- **Hyperledger Network**: A permissioned blockchain system based on Hyperledger Fabric, available on GitHub: Hyperledger-Network.



Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage
0.74% / 3200% (32 CPUs available)

Container memory usage
5.05GB / 7.47GB

Show charts

Search

Only show running containers

Delete

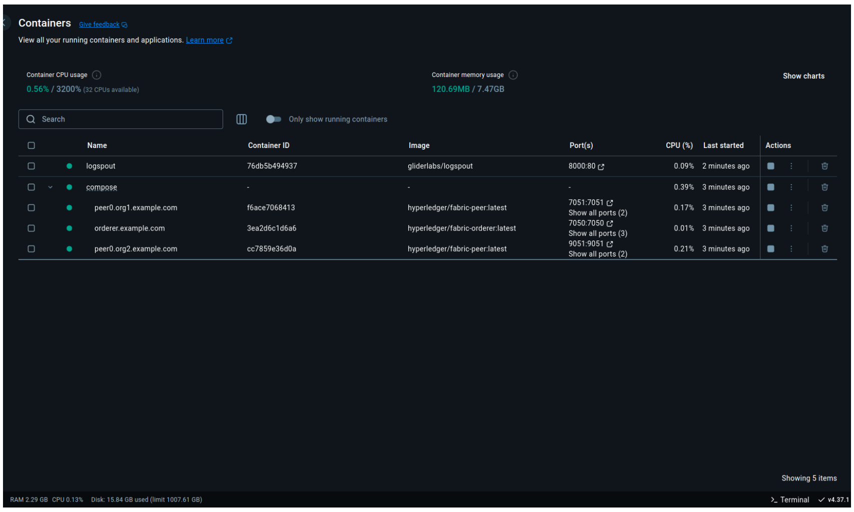
⊟		Name	Container ID	Image	Port(s)	CPU (%)	Last started
☑	●	banking_system	-	-	-	0.66%	5 minutes ago
☐	●	authentication-nwb	9eece94fa118	banking_system-authentication-nwbank	8082:8100	0.06%	5 minutes ago
☐	●	transaction-nwb	6bb2b9aa51ab	banking_system-transaction-nwbank	8084:8300	0.06%	5 minutes ago
☐	●	accounts-owb	85b223555569	banking_system-accounts-owbank	8087:8200	0.04%	5 minutes ago
☐	●	authentication-pwb	2d1ef3763692	banking_system-authentication-pwbank	8090:8100	0.05%	5 minutes ago
☐	●	accounts-pwb	4218cbd8defe	banking_system-accounts-pwbank	8091:8200	0.07%	5 minutes ago
☐	●	accounts-nwb	026dd664abc8	banking_system-accounts-nwbank	8083:8200	0.04%	5 minutes ago
☐	●	transaction-pwb	365c1257feef	banking_system-transaction-pwbank	8092:8300	0.06%	5 minutes ago
☐	●	authentication-owb	1735cf8e6246	banking_system-authentication-owbank	8086:8100	0.06%	5 minutes ago
☐	●	transaction-owb	d03c68aecff4	banking_system-transaction-owbank	8088:8300	0.06%	5 minutes ago
☐	●	clearingHouse	7389231bfd30	banking_system-clearinghouse	9091:9090	0.04%	5 minutes ago
☐	●	gateway-pwb	f9763ca09bae	banking_system-gateway-pwbank	8089:8080	0.04%	5 minutes ago
☐	●	gateway-nwb	cb17f23b4d9f	banking_system-gateway-nwbank	8081:8080	0.04%	5 minutes ago
☐	●	gateway-owb	ff0a11f684f3	banking_system-gateway-owbank	8085:8080	0.04%	5 minutes ago

Selected 1 of 14

RAM 7.06 GB CPU 0.03% Disk: 15.85 GB used (limit 1007.61 GB)

Terminal v4.37.1

Figure 3.1: Microservice Network Developed Using Docker Containers



Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage
0.56% / 3200% (32 CPUs available)

Container memory usage
120.69MB / 7.47GB

Show charts

Search

Only show running containers

☐		Name	Container ID	Image	Port(s)	CPU (%)	Last started
☐	●	logspout	76db5b494937	gliderlabs/logspout	8000:80	0.09%	2 minutes ago
☐	●	compose	-	-	-	0.39%	3 minutes ago
☐	●	peer0.org1.example.com	f6ace7068413	hyperledger/fabric-peer:latest	7051:7051 Show all ports (2)	0.17%	3 minutes ago
☐	●	orderer.example.com	3ea2d6c1d6a6	hyperledger/fabric-orderer:latest	7050:7050 Show all ports (3)	0.01%	3 minutes ago
☐	●	peer0.org2.example.com	cc7859e36d0a	hyperledger/fabric-peer:latest	9051:9051 Show all ports (2)	0.21%	3 minutes ago

Showing 5 items

RAM 2.29 GB CPU 0.13% Disk: 15.84 GB used (limit 1007.61 GB)

Terminal v4.37.1

Figure 3.2: Hyperledger Fabric Network Developed Using Docker Containers

The microservices system simulates real-world banking transactions with lightweight REST APIs and modular service design. The Hyperledger Fabric system utilizes smart contracts (chaincode) for secure asset creation and transfer, leveraging binaries and network setup configurations provided by the Hyperledger Fabric documentation. Special acknowledgment goes to the Hyperledger team for making the necessary binaries and documentation available for setting up the test network.

3.6 Load Testing with Custom Test Scripts

Custom test scripts were developed manually to simulate real-world transaction loads and analyze the performance of both systems under identical conditions:

- **Banking System:** REST API endpoints were stress-tested using scripts that issued concurrent transaction requests.
- **Hyperledger Fabric:** Smart contract invocations (asset transfers) were tested for performance under increasing network loads using custom scripts.

The scripts allowed granular control over transaction execution, logging, and error handling, enabling detailed insights into system behavior during peak loads.

Script snippet (Language: Javascript) for Load testing the hyperledger fabric
:

```
// -------------------------------------------------------------------
// Metrics
// -------------------------------------------------------------------
const avgUserCreation = userCreationTimes.reduce((a, b) => a + b, 0) /
    userCreationTimes.length;
const avgAssetCreation = assetCreationTimes.reduce((a, b) => a + b, 0) /
    assetCreationTimes.length;
const avgTransferTime = transferTimes.reduce((a, b) => a + b, 0) / transferTimes.length;
const avgQueryTime = queryTimes.reduce((a, b) => a + b, 0) / queryTimes.length;
const totalUsers = USERS.length;
const totalAssets = totalUsers * ASSETS_PER_USER;
const totalQueries = queryTimes.length;
console.log('\n=== Metrics ===');
console.log(`Number of users registered: ${totalUsers}`);
console.log(•
  ` Total user registration time: ${(
    userCreationTimes.reduce((a, b) => a + b, 0) / 1000
  ).toFixed(3)} s`
);
console.log(•
  ` Average user registration time: ${(avgUserCreation / 1000).toFixed(
    3
  )} s`
);
console.log(`\nNumber of assets created: ${totalAssets}`);
console.log(•
  ` Total asset creation time: ${(
    assetCreationTimes.reduce((a, b) => a + b, 0) / 1000
  ).toFixed(3)} s`
);
console.log(•
  ` Average asset creation time: ${avgAssetCreation.toFixed(2)} ms`
);
console.log(`\nNumber of transfers: ${totalAssets}`);
console.log(•
  ` Total transfer time: ${(
    transferTimes.reduce((a, b) => a + b, 0) / 1000
  ).toFixed(3)} s`
);
console.log•(` Average transfer time: ${avgTransferTime.toFixed(2)} ms`);
console.log(`\nNumber of queries: ${totalQueries}`);
console.log(•
  ` Total query time: ${(queryTimes.reduce((a, b) => a + b, 0) / 1000).toFixed(
    3
  )} s`
);
console.log•(` Average query time: ${avgQueryTime.toFixed(2)} ms`);
console.log('\nScript completed successfully!');
```

Script snippet (Language:Javascript) for testing the Springboot Microservices:

```
/******************************************************
 * Summarize Transaction Metrics
 ******************************************************/
function summarizeMetrics() {
  console.log('--- TRANSACTION METRICS ---');
  if (totalTransactions === 0) {
    console.log('No transactions processed.');
    return;
  }

  const averageTime = totalTime / totalTransactions;
  console.log(`Total transactions:     ${totalTransactions}`);
  console.log(`Total transaction time (ms): ${totalTime.toFixed(2)}`);
  console.log(`Average transaction time (ms): ${averageTime.toFixed(2)}`);

  const maxTime = Math.max(...transactionTimes);
  const minTime = Math.min(...transactionTimes);
  console.log(`Fastest transaction (ms): ${minTime.toFixed(2)}`);
  console.log(`Slowest transaction (ms): ${maxTime.toFixed(2)}`);
  console.log('---------------------------\n');
}
```

3.7 Metrics for Evaluation

The following metrics were used to evaluate and compare the systems:

- **Transaction Throughput:** Number of transactions processed per second.
- **Latency:** Time taken to complete a transaction.
- **Scalability:** Ability to handle increasing transaction loads.
- **Reliability:** Success rate of transactions without errors.
- **Traffic:** Number of users and accounts processed.
- **Security:** Traceability and auditability of transactions.

3.8 Data Analysis

Data from both systems were collected and analyzed to compare performance and scalability. For each test:

- **Banking System:** Log data was parsed to extract transaction times and outcomes.
- **Hyperledger Fabric:** Network logs were analyzed to evaluate block creation times, transaction validation, and endorsement latencies.

The results were visualized using graphs for time vs. throughput, activity spikes, and traffic handled.

3.9 Expected Outcome

The expected outcome was to identify the strengths and weaknesses of each system:

- **Banking System:** Expected to excel in transaction speed and throughput but face challenges in traceability and distributed trust.
- **Hyperledger Fabric:** Anticipated to provide robust security and traceability with higher transaction latency and complexity.

3.10 Limitations

1. **Hardware Constraints:**
    - Both systems were tested on a local workstation with limited resources.
    - Results may not scale linearly to enterprise-level hardware.
2. **Test Network Scope:**
    - The Hyperledger Fabric test network was simplified with five user nodes.
    - A production environment may involve more complex configurations and larger networks.
3. **Data Scope:**
    - Simulated data was used instead of live banking transactions, which may impact real-world applicability.

3.11 Conclusion

This chapter outlined the methodology used to develop, implement, and test two distinct banking architectures: microservices and Hyperledger Fabric. The use of custom test scripts for load testing and the metrics evaluated provided a comprehensive understanding of the systems' performance, scalability, and security. The subsequent chapters analyze the experimental results and discuss their implications for modern banking systems.

CHAPTER 4 :

ANALYSIS

4.1 Key Findings from Experiments

The experiments conducted on the two architectures—Banking Microservices and Hyperledger Fabric—provided quantitative insights into their performance and scalability:

1. **Transaction Speed:**

    - Banking Microservices demonstrated significantly faster transaction processing, averaging 39.17 milliseconds per transaction.
    - Hyperledger Fabric's average transaction time was 2017.95 milliseconds, attributed to the overhead of consensus mechanisms and block finality.

2. **Transaction Throughput:**

    - The microservices-based system successfully handled 43 transactions in 1684.23 milliseconds.
    - Hyperledger Fabric processed only 15 asset transfers within 30.269 seconds.

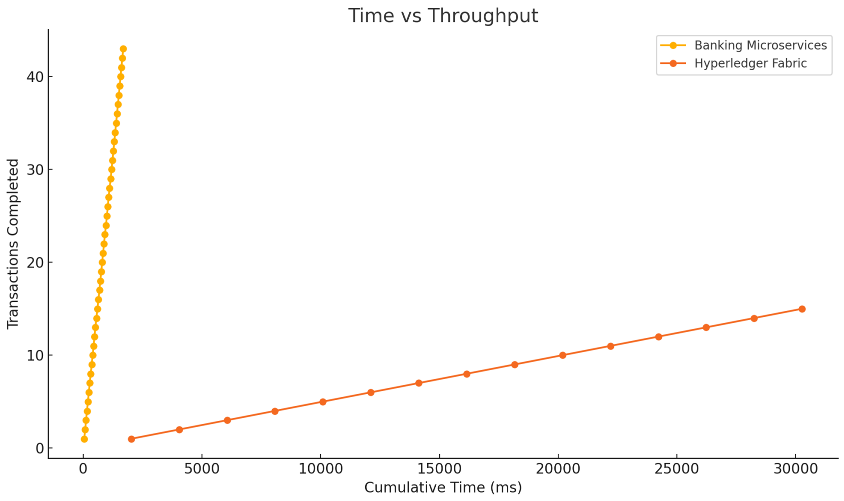


Figure 4.1: Time vs Throughput

3. **Reliability:**

   - Banking Microservices experienced minor issues with two failed transactions (HTTP 400 status), indicating possible issues with input validation or API orchestration.
   - Hyperledger Fabric executed transactions with 100% success, thanks to its robust consensus and validation model.

4. **Scalability:**

   - Microservices offer straightforward horizontal scaling through container orchestration tools such as Kubernetes.
   - Hyperledger Fabric's scalability is more complex due to consensus overhead and the need for additional peers and resources.

5. **Traffic Analysis:**

   - Banking Microservices handled 43 unique account transactions, representing individual users.
   - Hyperledger Fabric explicitly handled traffic for 5 created users, demonstrating its controlled and permissioned environment.

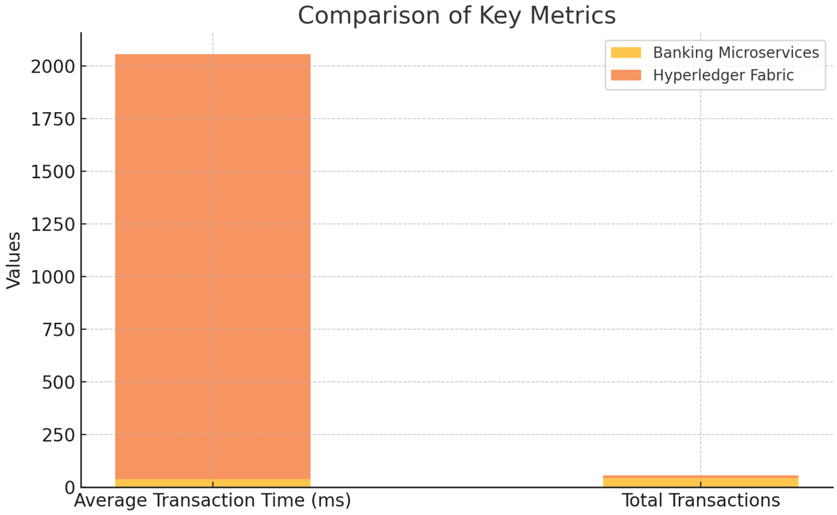


Figure 4.2: Average time vs Total Transactions

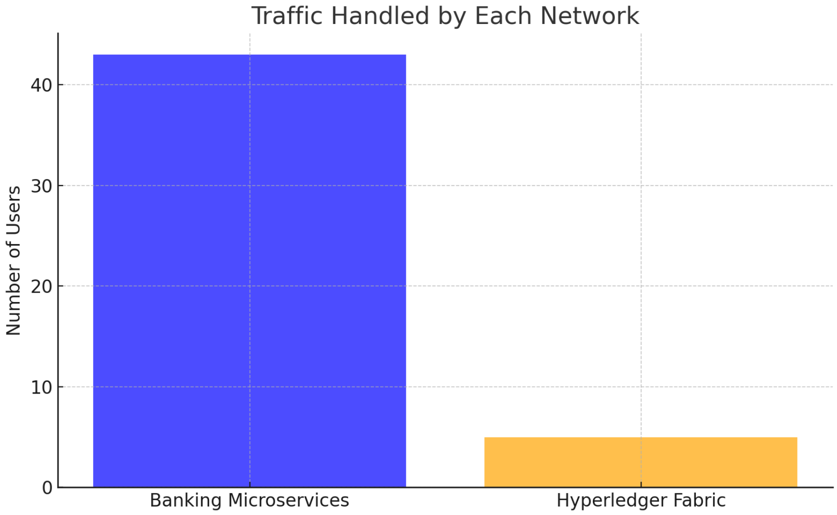


Figure 4.3: Total Traffic handled by each system

4.2 Performance Analysis

4.2.1 Banking Microservices:

- **Strengths:** Rapid transaction speed and high throughput.
- **Weaknesses:** Dependent on centralized trust and potentially vulnerable to partial system failures.

4.2.2 Hyperledger Fabric:

- **Strengths:** Immutable transaction records and distributed trust.
- **Weaknesses:** High transaction latency and complexity in setup.

4.3 Security and Auditability

4.3.1 Hyperledger Fabric

- Transactions are inherently auditable due to the immutable ledger structure.
- Each transaction is cryptographically signed and stored as part of the blockchain, ensuring traceability.
- Querying the ledger provides a complete history of asset ownership and transfers.

- Permissioned access allows detailed monitoring and role-based audit trails, facilitating compliance with regulations.

4.3.2 Banking Microservices

- Transactions are distributed across multiple databases, making auditing more complex.
- Without a centralized or immutable ledger, tracing a transaction's history requires querying multiple systems.
- Failures in inter-service communication or logging can lead to incomplete or missing transaction records.
- Requires additional layers of monitoring, logging, and integration to ensure traceability, which can be resource-intensive.

4.4 Comparative Metrics

A comparison of key metrics reveals the trade-offs between the two systems:

Metric	Banking Microservices	Hyperledger Fabric
Average Transaction Time	39.17 ms	2017.95 ms
Total Transactions Processed	43	15
Success Rate	95.35%	100%
Scalability	High	Moderate
Traffic Handled	43 Users	5 Users
Complexity	Low	High

Table 4.1: Comparison of key metrics

CHAPTER 5 :

RESULTS AND DISCUSSIONS

5.1 Discussion on Key Findings

5.1.1 Performance Suitability

- For high-throughput systems, such as real-time banking applications, Banking Microservices provide an edge due to their speed and ease of scaling.
- Hyperledger Fabric, despite its lower speed, is more suited for scenarios requiring strong guarantees of data integrity and trust.

5.1.2 Trade-offs

- The microservices approach offers simplicity and speed but lacks inherent mechanisms for distributed trust.
- Hyperledger Fabric's strengths lie in secure, tamper-evident data storage, which is critical for audit-heavy operations.

5.1.3 Real-World Application

- Banking Microservices would excel in customer-facing applications like online banking platforms.
- Hyperledger Fabric could be deployed in interbank settlements, trade finance, or cross-border payments.

5.2 Implications for Scalability and Security

5.2.1 Scalability

- Microservices: Easily scalable using container orchestration tools. Suitable for institutions with fluctuating workloads.
- Hyperledger Fabric: Requires careful planning for scaling due to its consensus mechanism. Additional nodes increase network complexity.

5.2.2 Security and Auditability

- Hyperledger Fabric ensures traceable, immutable, and secure transactions. Its permissioned network structure provides a built-in audit trail and cryptographic integrity.
- Banking Microservices require extensive monitoring and orchestration to achieve similar levels of traceability, often at higher operational costs.

CHAPTER 6 :

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

Blockchain technology is increasingly transforming various industries by offering decentralized, transparent, and secure solutions, especially within the enterprise and financial sectors. This study has examined the implementation and enhancement of blockchain networks, focusing on the technological frameworks offered by Hyperledger Fabric and their impacts on security, scalability, and performance. Private blockchain networks, such as those developed using Hyperledger Fabric, provide exceptional security and customization, ideally suited for applications demanding rigorous access controls and data integrity. The research conducted herein confirms the effectiveness of using modular architectures like that of Hyperledger Fabric for enterprise purposes, stressing its performance across different configurations, such as throughput and latency optimization. These results correspond to existing studies that highlight the efficiency of permissioned blockchains in shortening transaction cycles and enhancing data traceability. Conversely, Hyperledger's smart contract capabilities offer powerful tools for automating intricate business processes. However, issues such as scalability and low transaction speed pose challenges to its application in core banking services like online transactions and card payments. In applications where integrity of transaction and data security is more important like cross-border transactions, interbank settlements, delivery of financial products, and asset transfer is where Hyperledger offers robust security and transaction integrity. The application of blockchain underscores its transformative potential to improve security, transparency, and operational efficiency. However, obstacles to widespread adoption remain, notably regulatory uncertainties and interoperability issues.

6.2 Recommendations

6.2.1 Optimization of Blockchain Frameworks

- Enterprises should leverage modular blockchain frameworks like Hyperledger Fabric, which allow tailored configurations to optimize performance based on specific application requirements.

- Further research into alternative state databases (e.g., RocksDB and BadgerDB) should be conducted to enhance data retrieval and transaction processing speeds in permissioned blockchains.

6.2.2 Enhanced Security Protocols

- Develop advanced security mechanisms, such as Message Authentication Codes (MAC) and zero-trust architectures, to mitigate risks in private blockchain implementations.

- Conduct static code analysis for smart contracts to identify vulnerabilities and improve resilience against cyber threats.

6.2.3 Scalability Solutions

- Incorporate dynamic load-balancing schemes and network sharding techniques to address scalability bottlenecks in blockchain systems, particularly in Ethereum-based networks.

- Evaluate performance trade-offs between public and private blockchains to identify optimal solutions for specific use cases.

6.2.4 Adoption in Financial Systems

- Banks and financial institutions should adopt blockchain solutions for transaction management and fraud prevention, prioritizing platforms that offer secure consensus mechanisms and privacy.

- Leverage decentralized exchanges (DEX) and smart contracts to improve financial inclusion and transaction transparency.

6.2.5 Policy and Regulatory Alignment

- Engage policymakers to establish robust regulations that promote blockchain adoption while addressing privacy and compliance issues.
- Develop standardized frameworks to ensure interoperability and ease of integration across different blockchain platforms.

6.2.6 Future Research Directions

- Investigate the feasibility of integrating emerging technologies like AI and IoT with blockchain for enhanced decision-making and automation.
- Explore advancements in consensus algorithms to improve the sustainability of blockchain networks, focusing on reducing energy consumption.

6.3 Closing Thoughts

The findings and recommendations presented in this thesis underscore the transformative potential of blockchain technology in reshaping enterprise operations and financial systems. By addressing current limitations and leveraging advancements in blockchain frameworks, organizations can achieve greater efficiency, security, and scalability. The ongoing evolution of blockchain solutions promises a future where trustless, decentralized systems become integral to global digital ecosystems.

REFERENCES

Attia, Mohamed and Amira Hassan Abed (2024). "A Comprehensive Investigation for Quantifying and Assessing the Advantages of Blockchain Adoption in Banking Industry". In: *6th International Conference on Computing and Informatics, ICCI 2024*. Institute of Electrical and Electronics Engineers Inc., pp. 322–331. ISBN: 9798350373875. DOI: 10.1109/ICCI61671.2024.10485028.

Bashir, Imran (2023). "Chapter 14: Hyperledger". In: *Mastering Blockchain: Inner workings of blockchain, from cryptography and decentralized identities, to DeFi, NFTs and Web3, 4th Edition*. Fourth. Packt Publishing, pp. 443–468.

Bucchiarone, Antonio et al. (2018). "From Monolithic to Microservices: An Experience Report from the Banking Domain". In: *IEEE Software* 35.3, pp. 50–55. DOI: 10.1109/MS.2018.2141026.

Capuano, Roberta and Henry Muccini (Mar. 2022). "A Systematic Literature Review on Migration to Microservices: a Quality Attributes perspective". In: *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, pp. 120–123. DOI: 10.1109/ICSA-C54293.2022.00030.

Chang, Yao-Chung et al. (Feb. 2024). "A Private Blockchain System based on Zero Trust Architecture". In: *2024 26th International Conference on Advanced Communications Technology (ICACT)*, pp. 143–146. DOI: 10.23919/ICACT60172.2024.10471993.

Dimou, Stavros, Kostas Choumas, and Thanasis Korakis (2023). "On Using Hyperledger Fabric Over Networks: Ordering Phase Improvements". In: *2023 IEEE International Conference on Communications Workshops: Sustainable Communications for Renaissance, ICC Workshops 2023*. Institute of Electrical and Electronics Engineers Inc., pp. 440–445. ISBN: 9798350333077. DOI: 10.1109/ICCWorkshops57953.2023.10283642.

Fedorov, I. R. et al. (2021). "Blockchain in 5G Networks: Perfomance evaluation of private blockchain". In: *Wave Electronics and its Application in Information and Telecommunication Systems, WECONF - Conference Proceedings*. DOI: 10.1109/WECONF51603.2021.9470519.

Idris, Nur Farahi Binti et al. (2023). "Performance Analysis of Hyperledger Fabric on Multiple Infrastructure Setup". In: *2023 International Conference on Digital Applications, Transformation and Economy, ICDATE 2023*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9798350310689. DOI: 10.1109/ICDATE58146.2023.10248830.

Johnson, D, K Dandapani, and M Sharokhi (2024). "Blockchain Applications in Finance". In: *Blockchain Applications in Finance*.

Kaushal, Rajesh Kumar and Naveen Kumar (2023). "Blockchain Implementation with Hyperledger Fabric and Approach for Performance Evaluation". In: *2023 IEEE International Conference on Blockchain and Distributed Systems Security, ICBDS 2023*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9798350333763. DOI: 10.1109/ICBDS58040.2023.10346570.

Konkin, Anatoly and Sergey Zapechnikov (Jan. 2021). "Techniques for Private Transactions in Corporate Blockchain Networks". In: *Proceedings of the 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus 2021*. Institute of Electrical and Electronics Engineers Inc., pp. 2356–2360. ISBN: 9780738142753. DOI: 10.1109/ElConRus51938.2021.9396228.

Kuzlu, Murat et al. (July 2019). "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability". In: *Proceedings - 2019 2nd IEEE International Conference on Blockchain, Blockchain 2019*. Institute of Electrical and Electronics Engineers Inc., pp. 536–540. ISBN: 9781728146935. DOI: 10.1109/Blockchain.2019.00003.

Laishevskiy, Ivan, Artem Barger, and Vladimir Gorgadze (2023). "A Journey Towards the Most Efficient State Database for Hyperledger Fabric". In: *2023 IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2023*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9798350310191. DOI: 10.1109/icbc56567.2023.10174970.

Larsson, Magnus (2021). *Microservices with Spring Boot and Spring Cloud: Build resilient and scalable microservices using Spring Cloud, Istio, and Kubernetes*. Pakt Publishing.

Mallidi, Ravi Kiran, Manmohan Sharma, and Sreenivas Rao Vangala (2022). "Streaming Platform Implementation in Banking and Financial Systems". In: *2022 2nd Asian Conference on Innovation in Technology, ASIANCON 2022*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9781665468510. DOI: 10.1109/ASIANCON55314.2022.9909500.

Mathwale, Rupsingh (2023). "AHFD: A Framework for Deployment and Management of Hyperledger Fabric Enterprise Blockchain". In: *2023 International Conference on Data Science and Network Security, ICDSNS 2023*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9798350301595. DOI: 10.1109/ICDSNS58469.2023.10245496.

Muneeb, Muhammad et al. (2022). "SmartCon: A Blockchain-Based Framework for Smart Contracts and Transaction Management". In: *IEEE Access* 10, pp. 10719–10730. ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3135562.

Muslimin, Firnando, Ahmad Nurul Fajar, and Meyliana (Nov. 2020). "Business Process Management and Service Oriented Architecture Integration for Transactional Banking Application". In: *7th International Conference on ICT for Smart Society: AIoT for Smart Society, ICISS 2020 - Proceeding*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9780738143552. DOI: 10.1109/ICISS50791.2020.9307597.

Navarro, Antonio (2022). "Fundamentals of Transaction Management in Enterprise Application Architectures". In: *IEEE Access* 10, pp. 124305–124332. ISSN: 21693536. DOI: 10.1109/ACCESS.2022.3224759.

Padmavathi, U. et al. (2024). "Examining Architectural Aspects of Hyperledger Fabric: A Thorough Review". In: *2024 International Conference on Innovations and Challenges in Emerging Technologies, ICICET 2024*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9798350319019. DOI: 10.1109/ICICET59348.2024.10616336.

Patil, Kanchan, Dhanya Pramod, and Vijayakumar Bharathi S (2022). "Examining the intention to adopt Blockchain in the Banking Industry by combining Task-Technology-Fit and Technology Acceptance Theory". In: *2022 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems, ICETSIS 2022*. Institute of Electrical and Electronics Engineers Inc., pp. 217–224. ISBN: 9781665469197. DOI: 10.1109/ICETSIS55481.2022.9888946.

Quan, Brandon Liew Yi, Nur Haliza Abdul Wahab, Juniardi Nur Fadila, Yichiet Aun, S. K. L. Luk, et al. (May 2024). "The Frontier of Blockchain Privacy: Development of a Private Ethereum Network". In: *2024 IEEE 14th Symposium on Computer Applications Industrial Electronics (ISCAIE)*, pp. 117–122. DOI: 10.1109/ISCAIE61308.2024.10576280.

— (2024). "The Frontier of Blockchain Privacy: Development of a Private Ethereum Network". In: *14th IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2024*. Institute of Electrical and Electronics Engineers Inc., pp. 117–122. ISBN: 9798350348798. DOI: 10.1109/ISCAIE61308.2024.10576280.

Shah, Neelkumar K. et al. (2023). "Smart Contract Vulnerability Detection Techniques for Hyperledger Fabric". In: *2023 IEEE 8th International Conference for Convergence in Technology, I2CT 2023*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9798350334012. DOI: 10.1109/I2CT57861.2023.10126362.

Singh, Kuldeep et al. (2023). "Revolutionizing Digital Banking: Harnessing Blockchain Smart Contracts for Enhanced Security and Efficiency". In: *2023 3rd International Conference on Smart Generation Computing, Communication and Networking, SMART GENCON 2023*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9798350319125. DOI: 10.1109/SMARTGENCON60755.2023.10442642.

Sun, Qiucheng and Yuyu Yuan (2022). "GBCL: Reduce Concurrency Conflicts in Hyperledger Fabric". In: *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*. Vol. 2022-October. IEEE Computer Society, pp. 15–19. ISBN: 9781665410311. DOI: 10.1109/ICSESS54813.2022.9930267.

Taha, Marah Mohammed and Mafaz Alanezi (2023). “An optimized Byzantine Fault Tolerance Algorithm via MAC for Private Blockchain”. In: *Proceedings - International Conference on Developments in eSystems Engineering, DeSE*. Institute of Electrical and Electronics Engineers Inc., pp. 522–527. ISBN: 9798350381344. DOI: 10.1109/DeSE60595.2023.10468988.

Tressa, Neethu and C. Priya (2023). “Blockchain Based UPI Technology for Secured Peer-to-Peer Cryptocurrency Transactions”. In: *2023 Global Conference on Information Technologies and Communications, GCITC 2023*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9798350308167. DOI: 10.1109/GCITC60406.2023.10425797.

Vidwans, Shantanu et al. (2022). “Permissioned Blockchain Voting System using Hyperledger Fabric”. In: *2022 International Conference on IoT and Blockchain Technology, ICIBT 2022*. Institute of Electrical and Electronics Engineers Inc. ISBN: 9781665424165. DOI: 10.1109/ICIBT52874.2022.9807702.

Wan, Nianbin, Yuliang Liu, and Weidong Xiao (Oct. 2020). “A Financial Transaction Methods Based on MapReduce Technology and Blockchain”. In: *Proceedings - 2020 3rd International Conference on Smart BlockChain, SmartBlock 2020*. Institute of Electrical and Electronics Engineers Inc., pp. 109–113. ISBN: 9780738113630. DOI: 10.1109/SmartBlock52591.2020.00027.

Zand, Matt, Xun Wu, and Mark Anthony Morris (2021). *Hands-On Smart Contract Development with Hyperledger Fabric V2: Building Enterprise Blockchain Applications*. First. O’Reilly Media.

Zhang, Shenbin et al. (Sept. 2020). “Performance Diagnosis and Optimization for Hyperledger Fabric”. In: *2020 2nd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)*, pp. 210–211. DOI: 10.1109/BRAINS49436.2020.9223271.

Zhao, Zishan (2022). “Comparison of Hyperledger Fabric and Ethereum Blockchain”. In: *2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers, IPEC 2022*. Institute of Electrical and Electronics Engineers Inc., pp. 584–587. ISBN: 9781665409025. DOI: 10.1109/IPEC54454.2022.9777292.

APPENDIX A : Research Proposal

Link to Github Repository of Research Proposal

Please refer the Link for Details: Thesis Repository

APPENDIX B : Github Link of Code

Link to Github Repository of Code

Please refer the Link for Microservice & Hyperledger

APPENDIX C : Docker Compose Yaml

The Configuration file for the docker network

Since configuration files with secret key files are not uploaded on the internet, here is the complete docker-compose.yaml file for the springboot system:

```
   ############################################################################################################
# NW Bank Services

    #
################################################################################################################

services:
  gateway-nwbank:
    build:
      context: ./NWBank/NWB-API-Gateway-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: gateway-nwb
    volumes:
      - ./NWBank/NWB-API-Gateway-develop/target:/app/target
    environment:
      - SERVER_PORT=8080
      - IFSC_CODE=NWB0001
      -
          SECRET_KEY=ipm89FyJ5it9gIYB2E+FZM9il3e2mQtQsDqH/GBU4iJQF2lynN9xsM1vMSZmCHlpsUdzfIqvfqksVr4NoFTKpw==
    ports:
      - "8081:8080"
    networks:
      - nwnetwork
    depends_on:
      - authentication-nwbank
      - accounts-nwbank
      - transaction-nwbank

  authentication-nwbank:
    build:
      context: ./NWBank/NWB-Authentication-Service-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: authentication-nwb
    volumes:
      - ./NWBank/NWB-Authentication-Service-develop/target:/app/target
    environment:
      - SERVER_PORT=8100
      - IFSC_CODE=NWB0001
      -
          SECRET_KEY=ipm89FyJ5it9gIYB2E+FZM9il3e2mQtQsDqH/GBU4iJQF2lynN9xsM1vMSZmCHlpsUdzfIqvfqksVr4NoFTKpw==
      - ROOT_USER=admin
      - ROOT_PASSWORD=changeit
      - DB_URL=jdbc:mysql://host.docker.internal:3306/NWBank
      - DB_USERNAME=NWB_USER
      - DB_PASSWORD=PASS@123
      - DB_SCHEMA=NWBank
    ports:
      - "8082:8100"
    networks:
      - nwnetwork

  accounts-nwbank:
    build:
      context: ./NWBank/NWB-Accounts-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: accounts-nwb
    volumes:
      - ./NWBank/NWB-Accounts-develop/target:/app/target
    environment:
      - SERVER_PORT=8200
      - IFSC_CODE=NWB0001
      - DB_URL=jdbc:mysql://host.docker.internal:3306/NWBank
      - DB_USERNAME=NWB_USER
      - DB_PASSWORD=PASS@123
      - DB_SCHEMA=NWBank
    ports:
      - "8083:8200"
    networks:
      - nwnetwork

  transaction-nwbank:
    build:
      context: ./NWBank/NWB-Transaction-Service-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: transaction-nwb
    volumes:
      - ./NWBank/NWB-Transaction-Service-develop/target:/app/target
    environment:
      - SERVER_PORT=8300
      - IFSC_CODE=NWB0001
      - DB_URL=jdbc:mysql://host.docker.internal:3306/NWBank
      - DB_USERNAME=NWB_USER
      - DB_PASSWORD=PASS@123
      - DB_SCHEMA=NWBank
      - CLEARING_HOUSE_URL=http://clearinghouse:9090
    ports:
      - "8084:8300"
    networks:
      - nwnetwork


################################################################################################################
# OW Bank Services

    #
################################################################################################################

  gateway-owbank:
    build:
      context: ./OWBank/OWB-API-Gateway-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: gateway-owb
    volumes:
      - ./OWBank/OWB-API-Gateway-develop/target:/app/target
    environment:
      - SERVER_PORT=8080
      - IFSC_CODE=OWB0001
      -
          SECRET_KEY=YjWbDI1cSmUCAqTvlfdsKVhmeBjRV2DwpbBL7bynCxTKQeiNIQKr+1TeeXLlZkLeYoguNDhiQDOoy3kz2eHfbw==

    ports:
      - "8085:8080"
    networks:
      - ownetwork
    depends_on:
      - authentication-owbank
      - accounts-owbank
      - transaction-owbank

  authentication-owbank:
    build:
      context: ./OWBank/OWB-Authentication-Service-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: authentication-owb
    volumes:
      - ./OWBank/OWB-Authentication-Service-develop/target:/app/target
    environment:
      - SERVER_PORT=8100
      - IFSC_CODE=OWB0001
      -
          SECRET_KEY=YjWbDI1cSmUCAqTvlfdsKVhmeBjRV2DwpbBL7bynCxTKQeiNIQKr+1TeeXLlZkLeYoguNDhiQDOoy3kz2eHfbw==
      - ROOT_USER=admin
      - ROOT_PASSWORD=changeit
      - DB_URL=jdbc:mysql://host.docker.internal:3306/OWBank
      - DB_USERNAME=OWB_USER
      - DB_PASSWORD=PASS@123
      - DB_SCHEMA=OWBank
    ports:
      - "8086:8100"
    networks:
      - ownetwork

  accounts-owbank:
    build:
      context: ./OWBank/OWB-Accounts-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: accounts-owb
    volumes:
      - ./OWBank/OWB-Accounts-develop/target:/app/target
    environment:
      - SERVER_PORT=8200
      - IFSC_CODE=OWB0001
      - DB_URL=jdbc:mysql://host.docker.internal:3306/OWBank
      - DB_USERNAME=OWB_USER
      - DB_PASSWORD=PASS@123
      - DB_SCHEMA=OWBank
    ports:
      - "8087:8200"
    networks:
      - ownetwork

  transaction-owbank:
    build:
      context: ./OWBank/OWB-Transaction-Service-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: transaction-owb
    volumes:
      - ./OWBank/OWB-Transaction-Service-develop/target:/app/target
    environment:
      - SERVER_PORT=8300
      - IFSC_CODE=OWB0001
      - DB_URL=jdbc:mysql://host.docker.internal:3306/OWBank
      - DB_USERNAME=OWB_USER
      - DB_PASSWORD=PASS@123
      - DB_SCHEMA=OWBank
      - CLEARING_HOUSE_URL=http://clearinghouse:9090
    ports:
      - "8088:8300"
    networks:
      - ownetwork


################################################################################################################
# PW Bank Services

    #
################################################################################################################
  gateway-pwbank:
    build:
      context: ./PWBank/PWB-API-Gateway-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: gateway-pwb
    volumes:
      - ./PWBank/PWB-API-Gateway-develop/target:/app/target
    environment:
      - SERVER_PORT=8080
      - IFSC_CODE=PWB0001
      -
          SECRET_KEY=JT+b6WIcwHkVbx207Iofw/wWEQpvK+Mpa9+MCvmS3s+ELYXF6zf4YCHa10afQmT03USTDEyyWnX1C4n/5BEk4w==

    ports:
      - "8089:8080"
    networks:
      - pwnetwork
    depends_on:
      - authentication-pwbank
      - accounts-pwbank
      - transaction-pwbank

  authentication-pwbank:
    build:
      context: ./PWBank/PWB-Authentication-Service-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: authentication-pwb
    volumes:
      - ./PWBank/PWB-Authentication-Service-develop/target:/app/target
    environment:
      - SERVER_PORT=8100
      - IFSC_CODE=PWB0001
      -
          SECRET_KEY=JT+b6WIcwHkVbx207Iofw/wWEQpvK+Mpa9+MCvmS3s+ELYXF6zf4YCHa10afQmT03USTDEyyWnX1C4n/5BEk4w==
      - ROOT_USER=admin
      - ROOT_PASSWORD=changeit
      - DB_URL=jdbc:mysql://host.docker.internal:3306/PWBank
      - DB_USERNAME=PWB_USER
      - DB_PASSWORD=PASS@123
    - DB_SCHEMA=PWBank

  ports:
    - "8090:8100"
  networks:
    - pwnetwork

accounts-pwbank:
  build:
    context: ./PWBank/PWB-Accounts-develop
    dockerfile: Dockerfile
  extra_hosts:
    - "host.docker.internal:host-gateway"
  container_name: accounts-pwb
  volumes:
    - ./PWBank/PWB-Accounts-develop/target:/app/target
  environment:
    - SERVER_PORT=8200
    - IFSC_CODE=PWB0001
    - DB_URL=jdbc:mysql://host.docker.internal:3306/PWBank
    - DB_USERNAME=PWB_USER
    - DB_PASSWORD=PASS@123
    - DB_SCHEMA=PWBank
  ports:
    - "8091:8200"
  networks:
    - pwnetwork

transaction-pwbank:
  build:
    context: ./PWBank/PWB-Transaction-Service-develop
    dockerfile: Dockerfile
  extra_hosts:
    - "host.docker.internal:host-gateway"
  container_name: transaction-pwb
  volumes:
    - ./PWBank/PWB-Transaction-Service-develop/target:/app/target
  environment:
    - SERVER_PORT=8300
    - IFSC_CODE=PWB0001
    - DB_URL=jdbc:mysql://host.docker.internal:3306/PWBank
    - DB_USERNAME=PWB_USER
    - DB_PASSWORD=PASS@123
    - DB_SCHEMA=PWBank
    - CLEARING_HOUSE_URL=http://clearinghouse:9090
  ports:
    - "8092:8300"
  networks:
    - pwnetwork
  clearinghouse:
    build:
      context: ./NWB_ClearingHouse-develop
      dockerfile: Dockerfile
    extra_hosts:
      - "host.docker.internal:host-gateway"
    container_name: clearingHouse
    environment:
      - SERVER_PORT=9090
    ports:
      - "9091:9090"
    networks:
      - nwnetwork
      - ownetwork
      - pwnetwork
    depends_on:
      - transaction-nwbank
      - transaction-owbank
      - transaction-pwbank

networks:
  nwnetwork:
  ownetwork:
  pwnetwork:
```