INFORMATICS INSTITUTE OF TECHNOLOGY

INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER

PolyGov

Decentralized Governance through Multi-Chain Voting and Proposal Automation

A Dissertation by

Mr. Rayaan Rilshad

Supervised by

Mr. Sharmilan Somasundaram

Submitted in partial fulfilment of the requirements for the BSc Computer Science degree at the University of Westminster.

April 2025

ABSTRACT

The expansion of blockchain ecosystems and the rise of DAOs operating across multiple networks have introduced complex governance challenges. Frameworks like MULTAV require manual smart contract deployments for every proposal, lack reusable governance components, and struggle with scalability and automation. These challenges result in high operational overhead, reduced participation, and inefficient decision-making processes. A scalable and automated governance model is therefore essential to support secure and inclusive decision-making across fragmented blockchain environments.

To address these limitations, this research presents a system that introduces reusable smart contracts and a trusted off-chain relayer mechanism. The proposed architecture eliminates repetitive deployments by enabling dynamic proposal management through modular contracts. Proposals created on the main chain are automatically mirrored to secondary chains, and votes are cast and synchronized across chains via the relayer. The relayer aggregates finalized votes and submits them to the main chain for quorum validation and proposal execution. The system was developed using Solidity, TypeScript, Next.js and tested across EVM-compatible testnets like Arbitrum and BSC.

The system achieved more than 70%-unit test coverage, validating core functionalities such as proposal creation, voting synchronization, and cross-chain execution. Functional and non-functional tests confirmed reliable behaviour under network disruptions, unauthorized access attempts, and proposal state transitions. Evaluations by blockchain professionals acknowledged the system's practical utility, scalability, and modular design. Though limited to EVM testnets, PolyGov lays a strong foundation for scalable and secure decentralized governance across heterogeneous blockchain networks.

Subject Descriptors:

- Security and privacy → Cryptography → Symmetric cryptography and hash functions → Hash functions and message authentication codes
- Computing methodologies → Distributed computing methodologies → Distributed programming languages

Keywords: Governance, DAO, Multi-Chain, Cross-Chain, Interoperability, Voting

DECLARATION

I declare that the work presented in this dissertation is the result of my own independent research. It has not been submitted, in whole or in part, for any other academic degree or qualification at any other university or institution. All sources of information have been properly acknowledged, and credit has been given to the original authors where appropriate.

Student Name: K.G.M.N.F. Rayaan Rilshad

Registration Number: W1899330 | 20211315

Signature: ……. **Date:** 15th April 2025

ACKNOWLEDGEMENT

This research project has been a hard but very rewarding journey. During the months of research, development, testing, and writing, I faced many challenges that made me learn, change, and improve. And I could not have finished this project without the help, support, and advice from many people.

First, I would like to sincerely thank my supervisor, Mr. Sharmilan Somasundaram, for his constant guidance, useful advice, and helpful feedback at every step. His support helped me stay on track and motivated.

I am also grateful to my university lecturers for giving me the knowledge and skills I needed to complete this project. Their teaching was the base for much of my work. A special thank you to the Final Year Project module team, especially Mr. Saadh Jawwadh, for their lessons, advice, and support during the project.

I also want to thank my family and friends for always supporting and believing in me, especially when things got difficult. Their encouragement helped me keep going.

Furthermore, I would like to acknowledge the assistance of large language models (LLMs) and AI-based tools, including ChatGPT, Claude, Grok and others, which I used following the university guidelines. They helped me with idea generation, understanding concepts better, fixing grammar mistakes, giving feedback on structure, and solving code bugs. All the final research, analysis, and writing decisions were made by me.

Finally, I would like to thank everyone who helped me, directly or indirectly, during this project. This journey has taught me a lot, and I am truly thankful for all the support I received.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DAO	Decentralized Autonomous Organization
DeFi	Decentralized Finance
TxHash	Transaction Hash
dApp	Decentralized Application
UI	User Interface
UID	Unique Identifier

CHAPTER 01: INTRODUCTION

1.1. Chapter overview

This chapter introduces the background and context of the research, focusing on the limitations of current multi-chain governance systems in decentralized ecosystems. It outlines the specific problem being addressed, the motivation behind the study, and the gaps in existing work. The chapter also presents the research questions, objectives, and aim of the project, followed by the key challenges and contributions to both the problem and research domains.

1.2. Problem domain

Governance refers to the process of making decisions within a group, organization, or community (Cardoso, 2023). In traditional systems, especially within financial institutions, governance typically operates under formal hierarchies with centralized authority. In such models, only certain individuals or roles possess the power to initiate or approve decisions (Shah, 2024). While this approach can provide structure, it often leads to slow and complicated decision-making processes, especially when multiple layers of approval are involved (Hamid Ekal and Abdul-wahab, 2022). Moreover, traditional governance systems are criticized for their lack of transparency and heavy reliance on centralized control, limiting broader participation and accountability. (Hamid Ekal and Abdul-wahab, 2022).

To overcome these limitations of traditional governance models, decentralized technologies such as blockchain offer alternative frameworks that emphasize transparency, community participation, and trust minimization.

1.2.1. Blockchain governance

Blockchain technology introduces a decentralized alternative to traditional governance. This approach is commonly seen in Decentralized Finance (DeFi) systems and Decentralized Autonomous Organizations (DAOs), which aim to overcome the limitations of centralized governance. Through smart contracts and blockchain infrastructure, these systems promote transparency, fairness, and wider community participation in decision-making (Hamid Ekal and Abdul-wahab, 2022). In DAOs, token holders can vote on proposals and directly influence governance outcomes (Valiente and Pavón, 2024). Every proposal, vote, and decision is

immutably recorded on the blockchain, ensuring verifiability, transparency, and accountability (Bellavitis, Fisch and Momtaz, 2023).

1.2.2. Multi-chain governance

As the blockchain ecosystem grows, the number of independent blockchain networks continues to increase. This expansion has led many DAOs and decentralized systems to operate across multiple chains, creating new technical and governance challenges (Tan et al., 2024). One of the primary concerns in this environment is interoperability - the ability of different blockchain networks and dApps to communicate and function together effectively. This includes connecting blockchains with each other, linking dApps within and across chains, and integrating blockchain systems with external technologies (Belchior et al., 2021; Yadav et al., 2024).

In addition, governance in multi-chain ecosystems must address increased security risks. Projects with small market capitalizations and limited token supplies are particularly vulnerable to manipulation and governance-based attacks (Fan, Chai and Zhong, 2020). These issues highlight the need for decentralized multi-chain governance frameworks capable of synchronizing governance actions, enhancing security, and ensuring efficiency across blockchains.

Multi-chain governance systems aim to provide secure, scalable, and efficient frameworks for decision-making across blockchains. However, these systems introduce new challenges that require further research and innovation.

1.3. Problem definition

Blockchain-based governance systems, particularly those implemented through Decentralized Autonomous Organizations (DAOs), have introduced more transparent and inclusive decision-making models compared to traditional centralized governance (Hamid Ekal and Abdul-wahab, 2022; Shah, 2024). However, most blockchain governance frameworks, such as those used by Uniswap, Compound, and Aave, remain confined to single-chain environments and face challenges related to centralization, low participation, and interoperability (Ao et al., 2023; Feichtinger et al., 2023; Messias et al., 2024).

As blockchain ecosystems continue to expand, multi-chain governance has emerged as a critical need to coordinate decision-making across independent blockchains (Belchior et al., 2021; Yadav

et al., 2024). Interoperability between blockchains introduces new governance challenges such as managing proposals, voting processes, and security across decentralized systems (Sion et al., 2024).

While some work has addressed cross-chain governance, notably the Multi-Chain Token Backed Voting (MULTAV) framework (Fan, Chai and Zhong, 2020), major limitations persist. MULTAV introduces significant manual overhead by requiring the deployment of multiple governance smart contracts for every proposal across each participating blockchain. This process increases gas fees, requires high technical expertise, and is not scalable for systems with a growing number of proposals and chains (Fan, Chai and Zhong, 2020). Further, MULTAV does not implement reusable or modular smart contracts, resulting in redundant deployments and increased maintenance complexity.

Thus, despite the evolution of blockchain governance models, current multi-chain governance systems still lack efficient, automated, scalable, and reusable solutions for proposal management, voting coordination, and contract deployment. Addressing these limitations is essential to support secure, scalable, and effective governance across decentralized, multi-chain ecosystems

1.3.1. Problem statement

Current multi-chain governance systems lack automation, scalability, and reusable smart contracts, resulting in inefficient and complex cross-chain decision-making processes.

1.4. Research motivation

The author has been active in the cryptocurrency field since 2020 and has bought several DAO tokens on different blockchain networks. However, many of these tokens were on blockchains that did not support governance voting. To take part in voting, the author had to move (bridge) the tokens to the blockchain that allowed voting. This process was difficult, expensive, and took time. These challenges at times discouraged the author from participating in governance. This experience showed a major issue in how blockchain governance works today. It motivated the author to research how multi-chain voting can make it easier for people to vote across different blockchains without needing to move their tokens.

1.5. Research gap

A systematic review of blockchain governance frameworks (Liu et al., 2023) identified only one framework, the Multi-Chain Token Backed Voting (MULTAV) framework proposed by Fan, Chai and Zhong, (2020), as an approach supporting multi-chain governance. However, MULTAV suffers from critical limitations, including the need for manual deployment of smart contracts for each proposal, a lack of automation in proposal management and voting processes, and the absence of reusable smart contract components. These issues result in operational inefficiencies, scalability challenges, and high costs.

To date, there is no comprehensive solution that addresses these operational challenges through a fully automated, scalable, and reusable framework for proposal management in multi-chain governance systems. This research aims to fill that gap by designing a governance architecture that reduces manual effort, supports cross-chain coordination, and enhances contract efficiency.

1.6. Contribution to the body of knowledge

1.6.1. Contribution to the research domain

This research advances the academic field of blockchain governance by:

- Proposing an improved multi-chain governance framework that addresses the limitations of existing models such as MULTAV.
- Demonstrating how a trusted relayer mechanism can be securely integrated into a decentralized governance system to facilitate cross-chain proposal mirroring, voting synchronization, and finalization.
- Introducing a modular and reusable smart contract architecture that minimizes manual deployment overhead and supports dynamic governance processes.

1.6.2. Contribution to the problem domain

This research plans to offer practical solutions to real-world challenges in decentralized multi-chain governance by:

- Developing an automated system that allows proposals and voting to be mirrored, aggregated, and finalized securely across different blockchains without high operational overhead.

- Improving scalability and operational efficiency by eliminating the need for repeated manual smart contract deployments for each new governance action.
- Supporting more inclusive and accessible decentralized governance by simplifying user interactions and enabling low-barrier participation across multiple blockchain networks.

1.7. Research challenge

1. **Automation:** Automating the creation, deployment, and management of proposals across multiple chains requires advanced coordination and monitoring. This includes managing both on-chain and off-chain components, which adds to the technical difficulty.
2. **Cross-Chain Synchronization:** A key challenge is ensuring that proposal states and their outcomes remain consistent across different blockchain networks. Any delay or mismatch in synchronization can cause governance issues or lead to conflicting actions on separate chains.
3. **Security:** Governance systems, especially those in small DAOs, are at risk of attacks such as spam proposals or vote manipulation. It is important to protect communication channels and prevent unauthorized actions during cross-chain operations.
4. **Scalability:** As the number of proposals and blockchain networks increases, the system must scale efficiently without performance loss. This requires improving smart contract design, lowering transaction costs, and ensuring the relayer can manage the higher demand.

1.8. Research questions

RQ1: How can smart contracts be designed to manage multiple proposals dynamically without repeated deployments?

RQ2: What mechanisms can automate the deployment and management of proposals across multiple blockchain networks?

RQ3: How can cross-chain synchronization of governance activities be ensured in a secure and efficient manner?

1.9. Research aim

The aim of this research is to design and develop a scalable and automated multi-chain governance system that focuses specifically on proposal management and voting synchronization across multiple blockchain networks.

This research focuses on eliminating the inefficiencies in current decentralized DAO governance frameworks by introducing reusable smart contracts capable of managing multiple proposals dynamically. It also aims to automate the proposal lifecycle across multiple blockchain networks, thereby reducing manual overhead and operational complexity.

1.10. Research objectives

Objectives	Description	LOs Mapped	RQ Mapped
Problem Identification	**RO1:** To investigate the inefficiencies in current multi-chain governance systems, including manual deployment, lack of automation, and scalability issues.	LO1, LO4, LO8	RQ1
Literature Review	**RO2:** To review and critically evaluate existing governance models, frameworks, and automation strategies in blockchain systems. **RO3:** To identify limitations in current approaches regarding smart contract reusability, automation, and cross-chain synchronization.	LO1, LO3, LO4, LO5, LO8	RQ1, RQ2, RQ3
Requirement Analysis	**RO4:** To collect and document system requirements for a reusable, scalable, and automated multi-chain governance system. **RO5:** To define technical specifications for reusable smart contracts and cross-chain proposal workflows.	LO3, LO4, LO6	RQ1, RQ2, RQ3
System Design	**RO6:** To design a reusable smart contract architecture capable of managing multiple proposals dynamically.	LO1, LO2, LO4,	RQ1, RQ2, RQ3

	RO7: To design an automation mechanism for proposal deployment and lifecycle management across chains. **RO8:** To design a secure cross-chain synchronization method for governance activities.	LO6, LO7	
Implementation	**RO9:** To implement the reusable smart contracts and automation logic using appropriate blockchain frameworks. RO10: To integrate cross-chain synchronization via relayer or messaging protocols.	LO1, LO5, LO7	RQ1, RQ2, RQ3
Testing and Evaluation	**RO11:** To evaluate the scalability, automation efficiency, and cross-chain consistency of the proposed system.	LO4, LO7, LO8	RQ1, RQ2, RQ3
Documentation	**RO13:** To document the design, implementation, and evaluation of the proposed system with a critical reflection on outcomes and learning.	LO4, LO5, LO8	RQ1, RQ2, RQ3
Presentation	**RO14:** To present and defend the research work and system implementation in a viva voce examination.	LO9	RQ1, RQ2, RQ3

Table 1: Research Objectives

1.11. Chapter summary

In summary, this chapter provided an overview of the problem domain of decentralized multi-chain governance and identified inefficiencies in current systems, such as manual deployment and lack of automation. It explained the research motivation, defined the problem and objectives, and highlighted the academic and practical contributions of the study. The research questions and challenges were also outlined, setting the foundation for the following chapters.

CHAPTER 02: LITERATURE REVIEW

2.1. Chapter overview

This chapter reviews existing literature on blockchain governance, with a particular focus on proposal management and voting mechanisms in multi-chain governance environments. It examines various governance methods currently in use, identifies key challenges associated with multi-chain proposal deployment and voting synchronization, and critically evaluates existing solutions. Furthermore, the chapter reviews relevant technologies supporting multi-chain governance, discusses the limitations of current approaches, and outlines suitable evaluation and benchmarking methods to assess voting effectiveness and governance performance across multiple blockchain networks.

2.2. Concept map

The concept map offers a clear summary of the main topics discussed in this chapter. It highlights essential background information, key research areas, and the primary factors influencing the proposed system. The visual summary can be found in **Appendix A: Concept Map**.

2.3. Problem domain

Governance generally refers to the structures, systems, and processes through which decisions are made, implemented, and controlled within an organization or system (Hamid Ekal and Abdul-wahab, 2022). It involves setting rules, practices, and responsibilities to guide actions, ensure accountability, and facilitate collective decision-making (Liu et al., 2023).

2.3.1. Limitations of traditional governance systems

Traditional governance models, particularly in corporate, financial, and information technology domains, have several well-known limitations. These include centralized decision-making, rigid structures, and a general lack of transparency (Hamid Ekal and Abdul-wahab, 2022). In traditional financial systems, decision-making power often rests with a selected group of individuals or institutions, which reduces inclusiveness and responsiveness. This centralized control can make systems less adaptable and less representative of stakeholder interests (Hamid Ekal and Abdul-wahab, 2022).

Similarly, corporate governance models typically rely on strict hierarchies. While these structures may work in traditional organizations, they often fail to accommodate the fast-paced and decentralized nature of digital technologies like blockchain (Shah, 2024). In IT governance, many frameworks are designed for centralized organizations. These models are not suitable for decentralized systems, where decision-making needs to be shared across multiple, independent parties, with varying degrees of centralization (Liu et al., 2023).

Moreover, traditional governance frameworks struggle with the unique demands of blockchain environments. Blockchain systems often operate with autonomous code and rely on community-driven decision-making, which requires broad stakeholder consensus (Liu et al., 2023). These needs are difficult to meet within centralized or rigid governance models.

The global financial crisis in 2008 also revealed the weaknesses of traditional governance, especially its lack of accountability and transparency. This event led to greater interest in decentralized and more transparent alternatives (Hamid Ekal and Abdul-wahab, 2022). Additionally, researchers have pointed out that traditional structures often promote agency problems and conflicts of interest that are difficult to resolve within hierarchical systems (Saurabh, Rani and Upadhyay, 2024). In response to these limitations, blockchain-based governance models have emerged as promising alternatives designed to enhance transparency, participation, and trust in decision-making processes.

2.3.2. Emergence of blockchain-based governance

Blockchain technology offers new ways to think about governance. It provides tools and frameworks that can address many of the problems found in traditional systems (Cardoso, 2023; Saurabh, Rani and Upadhyay, 2024). One of the most important innovations is the Decentralized Autonomous Organization (DAO). DAOs use smart contracts, which are self-executing pieces of code which can be used to automate and decentralize decision-making. This increases transparency, allows more people to participate in the governance process, and reduces the risks linked to centralized control (Shah, 2024). For example, token-based voting lets users propose and vote on decisions, with outcomes enforced automatically by smart contracts. This allows a wide group of stakeholders to take part in the governance process.

To meet legal and regulatory needs, hybrid DAO models have also emerged. These models combine blockchain governance with features from traditional legal systems. Hybrid models aim to improve scalability, legal compliance, and protection against threats such as Sybil attacks, where one entity pretends to be multiple users (Shah, 2024). Blockchain governance systems are often modular, similar to open-source software. This modular design allows the governance rules and processes to be updated and improved over time depending on the situation (Cardoso, 2023).

In decentralized finance (DeFi), blockchain governance is critical for supporting quick upgrades, fixing bugs, and making inclusive decisions (Liu et al., 2023). Unlike traditional financial systems, which often operate behind closed doors, DeFi projects value openness and community involvement (Hamid Ekal and Abdul-wahab, 2022). DAOs also introduce the concept of algorithmic trust, where users trust the code instead of relying on intermediaries, which helps reduce human error and bias (Saurabh, Rani and Upadhyay, 2024).

While blockchain introduces a more inclusive and transparent model of governance, the way decisions are made, particularly through voting plays a crucial role in its success.

2.3.3. Role and models of voting in blockchain governance

Voting plays a central role in blockchain governance by allowing stakeholders to take part in decision-making processes. Several voting models have been developed to make governance fairer and reduce the concentration of power among a few users.

For example, in delegated voting, users can assign their voting rights to trusted representatives, which helps improve participation rates (Fritsch, Müller and Wattenhofer, 2024). Other models, such as lock-based voting and staking-based voting, give more voting power to users who commit their tokens for longer periods, encouraging long-term involvement (Fan, Chai and Zhong, 2020; Messias et al., 2024). Voting systems also differ in how voting power is distributed. Token-weighted voting is the most common model, but it often leads to the concentration of power in the hands of a few large token holders (Shah, 2024). To address this, alternative methods such as quadratic voting and reputation-based voting have been introduced. These aim to balance influence and promote fairer governance outcomes (Bellavitis, Fisch and Momtaz, 2023; Rikken, Janssen and Kwee, 2023).

While single-chain governance voting systems offer transparency and some automation, they also face several limitations and problems:

- **Governance Centralization:** A small number of large token holders often dominate voting outcomes, reducing decentralization and increasing the risk of biased decisions that favour major stakeholders (Ao et al., 2023).
- **Low Voter Participation:** High transaction fees, particularly on Ethereum, discourage smaller token holders from voting. As a result, participation rates remain low, and governance is often influenced by a limited group of active voters (Messias et al., 2024).
- **Lack of Interoperability:** Voting is usually limited to a single blockchain, preventing users from other chains from participating. This limits governance scalability in a multi-chain environment (Frangella and Herskind, 2022).
- **System Complexity:** Many governance systems require technical knowledge to participate effectively. Concepts like staking, delegation, and proposal thresholds can make governance difficult for casual users (Feichtinger et al., 2023).

These challenges point to the need for more inclusive, scalable, and interoperable voting systems, particularly as blockchain ecosystems expand beyond single-chain boundaries.

2.3.4. Governance challenges in multi-chain environments

As the blockchain industry continues to grow, different blockchains are being created to serve different purposes. This has led to a fragmented ecosystem where interoperability, the ability of blockchains to work together is becoming increasingly important (Feichtinger et al., 2023). Interoperability involves connecting one blockchain to another, enabling applications across chains to share data, and integrating blockchain systems with other technologies (Belchior et al., 2021; Yadav et al., 2024).

To address interoperability challenges, multi-chain architectures are being developed to manage communication and governance across different blockchain platforms (Sion et al., 2024). However, they also introduce new governance challenges, such as maintaining security, ensuring privacy, managing trust, and reaching consensus across independent chains.

Solving these problems often requires new types of organizational structures. One example is the master-slave model for cross-chain systems. In this setup, a central chain governs several smaller

chains using smart contracts. This approach has shown promise in areas such as energy systems (Sion et al., 2024).

Another important concern is the security of smaller blockchain projects. Projects with low market capitalization and limited token supply are more vulnerable to manipulation and governance attacks, where malicious actors can easily gain control of the system (Fan, Chai and Zhong, 2020). Multi-chain governance offers a solution by spreading authority across multiple, more established chains. The MULTAV framework is an example of a system that uses token-based voting across several blockchains to improve security and decision-making reliability (Fan, Chai and Zhong, 2020). However, current frameworks like MULTAV still face practical challenges such as high manual overhead, lack of automation in proposal management, and limited scalability.

In summary, as the blockchain ecosystem becomes more complex, the need for better governance solutions is growing. Multi-chain governance models must not only support interoperability but also ensure that decisions are made securely, fairly, and efficiently across a range of decentralized systems.

2.4. Existing work

2.4.1. Single chain governance voting

Single-chain governance voting refers to decentralized decision-making systems that operate within a single blockchain ecosystem. In this model, all voting actions and governance decisions are recorded on-chain, providing a high level of transparency and security (Ao et al., 2023).

However, single-chain governance models limit participation to users within that specific blockchain, restricting interoperability and scalability (Ao et al., 2023). Several major decentralized finance (DeFi) protocols, such as Uniswap, Compound, and Aave, use single-chain governance voting systems. While these systems demonstrate strengths in decentralization, voter participation, and governance efficiency, they also show clear limitations when addressing broader, cross-chain governance needs.

DataDAO (Yadav et al., 2024)

DataDAO Club is an investment management platform that implements on-chain governance using smart contracts and Filecoin-based storage to facilitate decentralized decision-making. The governance framework operates entirely within a single blockchain, ensuring transparency and security but restricting participation to users within that specific ecosystem.

DataDAO Club employs an on-chain governance model in which decision-making processes, such as proposal submission and voting, are executed through smart contracts. The platform supports various voting mechanisms, including token-based voting, delegated voting, and quadratic voting. This governance structure offers key advantages, particularly in terms of transparency and security. Since all governance decisions are recorded on-chain, they remain auditable and resistant to tampering. Additionally, the direct execution of governance decisions within the blockchain minimizes reliance on off-chain mechanisms. The platform also ensures decentralized data management by leveraging Filecoin, which helps maintain the security and integrity of governance-related data.

However, the platform has certain limitations. One major challenge is lack of interoperability, as governance is confined to a single blockchain, preventing token holders from other blockchain ecosystems from participating. Additionally, governance fragmentation poses a challenge for projects operating across multiple blockchains, as they cannot coordinate governance decisions effectively under this model.

Uniswap

Uniswap's governance is structured around a Decentralized Autonomous Organization (DAO), where UNI token holders can vote on protocol changes, treasury allocations, and feature upgrades (Fritsch, Müller and Wattenhofer, 2024). A key advantage of Uniswap’s governance model is its full on-chain transparency, which ensures auditability and censorship resistance (Messias et al., 2024). Additionally, liquid democracy allows token holders to delegate their voting power, providing governance flexibility (Shah, 2024).

However, Uniswap faces significant governance challenges. Studies show that less than 10% of UNI tokens are actively used in governance votes, and the top 10 token holders control 44.72% of the voting power, leading to concerns about centralization and reduced decentralization

(Eisermann et al., 2025). Moreover, high Ethereum gas fees discourage smaller token holders from voting, further limiting inclusivity and participation (Feichtinger et al., 2023).

Compound

Compound was one of the first major DeFi protocols to introduce token-based governance, influencing many subsequent governance models, including Uniswap's (Messias et al., 2024). Governance is controlled by COMP token holders, who can propose and vote on protocol decisions, including interest rate adjustments, risk parameter modifications, and asset listings (Fritsch, Müller and Wattenhofer, 2024).

A key strength of Compound's governance system is its structured decision-making process, which includes formal community discussions, quorum thresholds, and timelocks for the execution of approved proposals (Eisermann et al., 2025). However, Compound also suffers from significant centralization issues. Studies have shown that as few as three voters can control 50% or more of governance decisions, mirroring the centralization problems seen in Uniswap (Messias et al., 2024). As with Uniswap, high Ethereum gas fees present a major barrier to participation, particularly for smaller token holders (Feichtinger et al., 2023).

AAVE

Aave's governance operates through AAVE token holders, who can vote on risk parameters, protocol upgrades, and asset listings (Frangella and Herskind, 2022). Unlike Compound and Uniswap, Aave introduces a dual-token system, using stkAAVE (staked AAVE), which provides staking rewards while also serving as a governance token (Ao et al., 2023). This approach aims to align incentives by rewarding active participants while strengthening protocol security. Aave governance has also expanded through multi chain deployments (separate governance on different chains); however, governance decisions remain largely Ethereum-centric, limiting true interoperability across chains (Frangella and Herskind, 2022).

Despite its innovations, Aave faces challenges similar to other DeFi governance models. Empirical studies show that a small number of delegates control most governance votes, raising concerns about centralization (Ao et al., 2023). Additionally, Aave's governance system is relatively complex, with a multi-layered proposal structure that creates a steep learning curve for users. This

complexity discourages casual participants and reduces overall community involvement (Feichtinger et al., 2023).

Protocol	*Type*	*Voting*	*Who can vote?*	*Delegation*	*Voting Aggregation*	*How proposals are implemented?*
AAVE [2]	Lending	on-chain	addresses with delegated tokens	yes	on-chain	on-chain via an SC call.
Balancer [10]	DEX	off-chain	stakers with locked tokens	yes (off-chain)	off-chain	via 6-of-11 multisig.
Compound [55]	Lending	on-chain	addresses with delegated tokens	yes	on-chain	on-chain via an SC call.
Convex Finance [25]	Yield Farming	off-chain	stakers with locked tokens	yes (off-chain)	off-chain	via 3-of-5 multisig.
Curve [27]	DEX	on-chain	stakers with locked tokens	yes	on-chain	on-chain through an SC call.
Maker Executive [59]	Stablecoin	on-chain	holders	no	on-chain	New Governance Contract requires more MKR staked than previous.
Maker Polling [59]	Stablecoin	on-chain	addresses with delegated tokens	yes	off-chain	Engineers at Maker create the governance contract based on the voting outcome.
Uniswap [3]	DEX	on-chain	addresses with delegated tokens	yes	on-chain	on-chain via an SC call.
1Inch [1]	DEX Aggregator	off-chain	stakers with locked tokens	yes	off-chain	via 7-of-12 multisig.
Ampleforth [5]	Stablecoin	on-chain	addresses with delegated tokens	yes	on-chain	on-chain via an SC call.
Rarible [70]	NFT	on-chain	stakers with locked tokens	yes	on-chain	on-chain via an SC call. However, a 3-seat security council has veto power over proposals that they deem malicious.

Figure 1: A comparison of voting mechanisms (Messias et al., 2024)

A summary of governance mechanisms across several leading DeFi protocols is shown in here and it highlights that most existing governance frameworks rely heavily on single-chain, on-chain voting models, with limited support for true multi-chain environment. Furthermore, several protocols use off-chain voting aggregation and multisig-based proposal implementation, which introduces centralization risks and reduces transparency compared to fully on-chain systems.

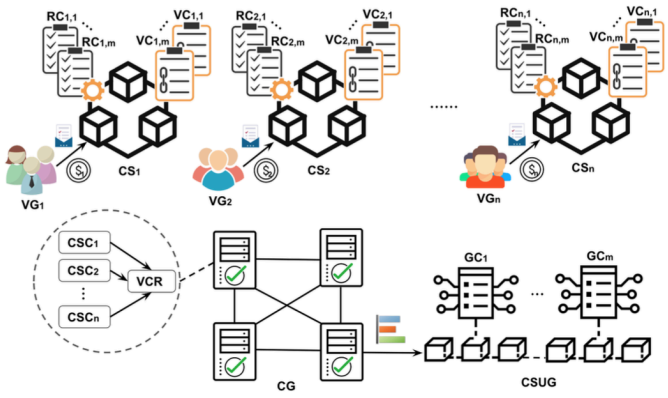
2.4.2. Multi chain governance

According to Liu et al., (2023), cross-chain governance and voting remain under-researched areas within blockchain governance studies. The Multi-Chain Token Backed Voting (MULTAV) framework proposed by Fan, Chai and Zhong, (2020) is one of the very few research works that directly address multi-chain governance voting.

MULTAV Framework (Fan, Chai and Zhong, 2020)

Multi-chain governance is a decentralized approach that allows decision-making processes to span multiple independent blockchain networks. The MULTAV framework introduces a model where token holders from various blockchains can participate in governance for a new blockchain project. By involving participants from different ecosystems, the framework aims to reduce governance manipulation risks and enhance security, particularly for emerging blockchain systems.

In the MULTAV framework, a governance proposal (GP_j) requires deploying a governance smart contract (GC_j) on the target blockchain, as well as registration ($RC_{i.j}$) and voting smart contracts ($VC_{i.j}$) on each participating blockchain. The voting results from these smart contracts are collected using Ethereum light clients and processed through a vote counting routine (VCR) to determine the final decision. While this structure enhances security by distributing governance power across multiple chains, it also creates significant challenges for proposal initiators, who must manually deploy new contracts for each proposal across all participating blockchains.



Although MULTAV presents an innovative solution for multi-chain governance, its practical implementation has several key limitations:

- **Manual Deployment Overhead:** MULTAV's reliance on manual contract deployment is one of its most notable inefficiencies. Each governance proposal requires a new set of smart contracts to be deployed across multiple blockchains. This approach Increases gas fees, requires technical expertise and creates redundant contract deployments
- **Scalability Constraints:** The multiplicative nature of contract deployments in MULTAV presents a serious scalability challenge. If there are m proposals and n blockchains, the total number of smart contracts required is $m \times n$. This results in high computational and

storage overhead, making it harder to track and manage contracts across multiple chains and growing inefficiencies in governance processes

- **Lack of Reusable Smart Contracts:** MULTAV does not implement modular or upgradeable smart contracts. Each proposal requires a completely new deployment, even if the governance rules remain the same. This leads to contract redundancy, slower execution, and increased maintenance.

As a result, MULTAV's dependence on manual deployment, lack of scalability, and absence of reusable governance components create inefficiencies that limit its practical application. Addressing these limitations is critical to making multi-chain governance more efficient, scalable, and usable for real-world blockchain ecosystems.

2.5. Technological review

2.5.1. Blockchain technology

Blockchain is a decentralized and distributed digital ledger that records transactions across multiple computers (Nakamoto, 2008). This architecture keeps data secure, transparent, and immutable in the blockchain. Unlike traditional centralized systems, where a single authority is responsible for managing and verifying data, blockchain relies on a network of participants, known as nodes, to maintain and validate the ledger through consensus mechanisms. This decentralized structure eliminates single points of failure and significantly enhances system security (Marella et al., 2020).

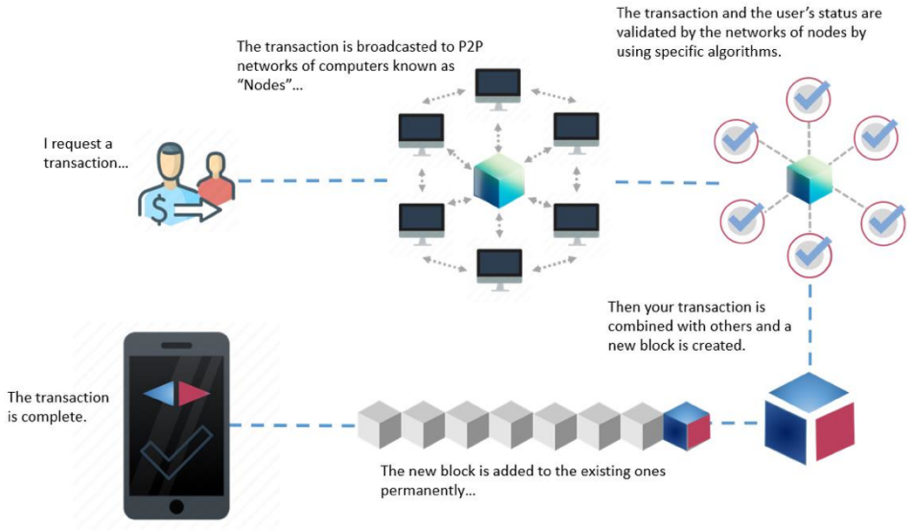


Figure 2: Blockchain Sample Workflow (Taherdoost, 2022)

As a result, blockchain technology provides a robust foundation for secure governance systems, making it particularly suitable for decentralized decision-making environments where transparency, trust, and immutability are critical.

2.5.2. Smart contracts

Smart contracts are self-executing programs deployed on blockchain networks that automatically carry out predefined instructions when triggered by transactions (Valiente and Pavón, 2024). These programs act as extensions of the blockchain's state and typically run within environments such as the Ethereum Virtual Machine (EVM), which supports Ethereum and other compatible blockchains (Belchior et al., 2021).

In governance systems, smart contracts play a critical role by automating rules, managing governance data, and enforcing voting outcomes on-chain (Liu et al., 2023). Their key characteristics include self-execution, immutability, transparency, and the ability to track internal state changes. Governance frameworks often encode decision-making procedures, such as voting mechanisms and quorum requirements, directly into smart contracts, as commonly seen in DAOs (Mohammed Abdul, Shrestha and Yong, 2024).

Due to their automated and tamper-resistant design, smart contracts are essential for enabling trustless and scalable governance. In multi-chain environments, smart contracts can be used to dynamically manage proposals, synchronize voting processes across chains, and ensure the reliable execution of governance decisions (Fan, Chai and Zhong, 2020).

2.5.3. Blockchain governance mechanisms

2.5.3.1. On-Chain governance

On-chain governance refers to decision-making processes that are directly embedded in the blockchain protocol and executed via smart contracts (Liu et al., 2023). When a proposal is made, token holders vote directly on the blockchain, and if the proposal meets the predefined requirements (e.g., a majority vote and reaching a quorum), the changes are automatically implemented by the smart contracts (Liu et al., 2023; Falk et al., 2024). Aave, Compound, and Uniswap utilize on-chain voting mechanisms for protocol upgrades, parameter adjustments, and other key decisions (Fritsch, Müller and Wattenhofer, 2024).

2.5.3.2. Off-Chain governance

Off-chain governance encompasses decision-making processes that occur outside of the blockchain. This often involves discussions in community forums, social media platforms, and through informal coordination among stakeholders (Liu et al., 2023). Proposals are typically discussed and refined off-chain before potentially being brought to an on-chain vote for formal approval (Falk et al., 2024). The outcomes of off-chain processes may then be implemented through on-chain actions, often via multi-signature contracts or by core development teams. Balancer and Convex Finance use Snapshot for off-chain voting, with the results then potentially enacted on-chain (Messias et al., 2024).

2.5.4. Voting mechanism

Blockchain-based governance systems use a variety of voting mechanisms to determine how decisions are made and who gets to influence them. These mechanisms can be categorized into how voting is conducted (voting methods) and how voting power is distributed (voting power models).

2.5.4.1. Voting methods

These mechanisms describe how votes are cast and who is eligible to vote or represent others.

a. Delegated Voting (Liquid Democracy)

Delegated voting allows token holders to either vote directly or assign their voting power to a delegate. Delegates can vote on their behalf, and delegation can be changed at any time (Fritsch, Müller and Wattenhofer, 2024). This model promotes active representation and can enhance participation.

b. Lock Voting

In this model, voting power is earned by locking tokens for a fixed period. The longer the duration and the greater the quantity of locked tokens, the more voting power is received. This discourages short-term speculation and rewards long-term commitment (Messias et al., 2024).

c. Staking-Based Voting

Voting rights are granted to users who stake their tokens in the network. Voting power is typically proportional to the amount and duration of the stake. This model ties governance to network participation and economic security (Fan, Chai and Zhong, 2020).

2.5.4.2. Voting power models

These mechanisms define how much influence each participant has when voting.

a. Token-Weighted Voting

Token-weighted voting is the most common method in DAOs, where the influence of a member's vote is directly proportional to the number of governance tokens they hold. This system carries the risk of centralizing power, as a small group of large token holders ("whales") can dominate the outcome (Shah, 2024).

b. Quadratic Voting

Quadratic voting is a variation of token-based voting that attempts to balance the influence between large and small token holders (Bellavitis, Fisch and Momtaz, 2023). The voting power a member receives is proportional to the square root of the number of tokens they commit (Shah, 2024). While it helps prevent major power concentration, it is still vulnerable to Sybil attacks, where a single entity creates multiple fake identities to gain disproportionate influence (Shah, 2024).

c. Reputation-Based Voting

Here, voting power is based on a participant's reputation, not their token balance. Reputation is earned through meaningful contributions to the DAO, such as proposals, development, or moderation (Rikken, Janssen and Kwee, 2023). This model is used to empower contributors who may not be financially wealthy but are essential to the DAO's health.

d. Identity-Based Voting

Identity-based voting seeks to ensure that each unique individual has a certain amount of voting power, often to mitigate Sybil attacks in systems like quadratic voting. This can involve using identity verification platforms to ensure that each participant is only allowed one verified wallet (Shah, 2024).

2.5.5. Cross-chain communication technologies

In multi-chain voting systems, seamless communication between independent blockchain networks is essential to synchronize proposals, voting processes, and execution outcomes. Cross-chain communication technologies provide the necessary infrastructure to coordinate state and data across chains, ensuring consistency, security, and reliability in governance activities. This section reviews major technologies used for cross-chain communication and evaluates their suitability for multi-chain voting environments.

Property	Notary mechanism	Sidechain/relay	Hash-locking	Distributed private key control	Notary scheme + sidechains mixing technology
Principle of realization	Trust a group of notaries	Collect the information of the original chain and verify it	Validate hash-locks and time-locks	Separation of ownership and use rights of assets	Use notaries for transaction authentication, and use sidechain for Token locking
Interoperability	All	All	Cross-dependence	All	All
Trust model	Most notaries are honest	The chain will not fail	The chain will not fail	The chain will not fail	Mixed model
Number of participating chains	Multi-chain	Double-chain/multi-chain	Double-chain	Multi-chain	Double-chain/multi-chain
Cross-chain asset transfer	Support	Support	Not support	Support	Support
Cross-chain asset mortgage	Support	Support	Most support	Support	Support
Applicable cross-chain primitives	Support	Support	Do not directly support	Support	Support
Multi-currency smart contracts	Difficult	Difficult	Not support	Support	Difficult
Cost ratio	Medium	High	Medium	Medium	Low
Security	Low	Low	Medium	Medium	High
Transaction speed	Slow	Slow	Medium	Medium	Fast
Difficulty to achieve	Medium	Medium	Easy	Medium	Difficult
Limitation	Relying on a third-party notary	Adapt to many scenarios, absolute consistency of access chain	The scene is single, the initiator has the initiative	The scope of application is small, and the smart contract needs to be improved	Difficult to implement multi-currency smart contracts

Figure 3: Cross-chain mechanism comparison (Ou et al., 2022)

1. Notary Mechanism

The notary mechanism relies on trusted intermediaries, known as notaries, to validate and transfer transactions between blockchains. In this approach, even if some notaries are compromised, groups of other notaries can verify transactions through multi-step protocols, maintaining security and efficiency (Chen et al., 2023). While notary mechanisms offer a relatively secure and efficient method for cross-chain interactions without significantly impacting transaction time or costs, they introduce a degree of centralization (Xiong et al., 2022) which may not suitable for the multi-chain voting system comparing to other methods.

2. Sidechain

Side chains are like independent blockchains that operate in parallel to a primary blockchain, which letting assets and data can move between with them. In this case scalability and flexibility is achieved, in part, through the allowance of each sidechain to run on custom consensus rules. This method is especially helpful for applications that have features or costs that need to isolate (Back et al., 2014). However, in the context of multi-chain governance, sidechains are often tightly coupled to their primary chains and lack compatibility with external networks (Ou et al., 2022). As a result, they are less effective for governance systems that need to coordinate across heterogeneous blockchain environment.

3. Relay

Relayers operate by monitoring events on a source blockchain and submitting relevant data or transactions to a destination chain. This allows real-time state tracking and cross-chain synchronization without requiring trust in a centralized party. In multi-chain voting systems, a relayer can efficiently mirror proposals from the main governance chain to secondary chains, collect votes across networks, and relay final vote tallies back for execution. While relayers can introduce development complexity, they offer a trust-minimized and modular solution for decentralized governance (Ou et al., 2022). Relayers enable deterministic, rule-based communication between chains, ensuring secure propagation of proposals and votes. By balancing decentralization, flexibility, and practical implementation, relay mechanisms are considered the most appropriate choice for supporting scalable and secure multi-chain governance systems.

4. Hash-Locking

Hash-locking is commonly used in atomic swaps to ensure that cross-chain transactions occur simultaneously while maintaining atomicity. This is achieved through Hash Time Lock Contracts (HTLCs), where assets are locked on one blockchain and can only be unlocked on another blockchain upon the presentation of a valid hash pre-image (Bhatia, Jain and Singh, 2021).

Although hash-locking provides a secure method for cross-chain asset transfers, it requires both parties to remain synchronized during the transaction process. This dependency can lead to delays and increased complexity if either party becomes unresponsive or needs to rejoin the network. As a result, hash-locking is not well-suited for multi-chain governance systems, which require continuous, asynchronous, and scalable proposal management and voting coordination.

5. Notary + Sidechain

Notary + Sidechain hybrid approach achieved security and scalability by combining the notary mechanism with side chains. The transaction validity is ensured with the notary mechanism, and scalability and flexibility are gained by side chains (Hardjono, Lipton and Pentland, 2018). In combining these two methods, this addresses the some of the limitations of each method alone. However, while this combination can enhance transaction throughput, it still introduces partial centralization through notary reliance and may not fully meet the decentralization and interoperability requirements of multi-chain governance systems.

In summary, while several cross-chain communication methods exist, the relayer mechanism offers the best trade-off between decentralization, efficiency, and compatibility for multi-chain governance voting systems. Relayers enable event-based synchronization, minimize trust assumptions, and align well with the dynamic requirements of cross-chain proposal handling and voting aggregation.

2.5.6. Supporting tools and frameworks

OpenZeppelin

OpenZeppelin is a widely used open-source framework designed to support the development of secure and efficient smart contracts on the Ethereum blockchain. It provides modular libraries, such as SafeMath, and standard contract implementations like ERC20, which help developers create reliable and secure decentralized applications (Pierro and Tonelli, 2021; Liu et al., 2024).

Given the high security requirements of smart contracts, developers frequently rely on OpenZeppelin's well-tested and audited codebase. The framework is employed across various blockchain applications, including cross-chain token transfers (Guo et al., 2024) and decentralized voting systems, where upgradable proxy contracts facilitate long-term maintenance and adaptability (Saim et al., 2022).

Overall, OpenZeppelin plays a critical role in Ethereum smart contract development by offering trusted tools and standardized design patterns that enhance security, maintainability, and functionality (Gec et al., 2023). Its robust contract libraries are particularly valuable for governance applications, where the reliability and upgradeability of voting and proposal management contracts are essential for sustainable multi-chain governance systems.

Inter Planetary File System (IPFS)

The Inter Planetary File System (IPFS) is a decentralized storage protocol that enables blockchain applications to store and share data off-chain. In multi-chain governance systems, IPFS can be utilized to store proposal metadata, voting records, or other governance-related information in a distributed and tamper-resistant manner, thereby reducing on-chain storage requirements (Belchior et al., 2021). Its shared data layer allows different blockchains to reference the same content using cryptographic hashes, supporting consistent and verifiable cross-chain access.

By offloading non-critical governance data, IPFS enhances scalability and performance in multi-chain architectures, particularly in systems involving sidechains or consortium blockchains (Sion et al., 2024). In multi-chain voting environments, IPFS complements relayers by managing off-chain content, ensuring that governance processes remain efficient, cost-effective, and resistant to censorship or tampering.

2.6. Evaluation and benchmarking

Existing research on blockchain-based governance provides valuable insights into voting mechanisms but often lacks formal benchmarking metrics. Furthermore, due to the decentralized and immutable nature of smart contracts, benchmarking decentralized governance systems presents unique challenges, including dynamic system behaviour, varying network conditions, and upgrade constraints. Despite these difficulties, several core evaluation criteria are commonly used to assess the effectiveness of governance systems.

1. **Security and Vulnerability:** Smart contracts must be secure against exploits that could compromise governance integrity. Vulnerabilities in contract logic, such as those seen in the Ethereum DAO incident, highlight the need for formal verification and thorough audits to prevent manipulation or unauthorized control (Liu et al., 2023).
2. **Voting Power Distribution:** Evaluating smart contract implementations includes analysing how different voting models, such as delegated voting or quadratic voting affect power distribution and voter influence (Fritsch, Müller and Wattenhofer, 2024).
3. **Fairness and Accuracy:** Governance contracts must enforce voting logic as intended, without allowing manipulation. This includes validating that all rules like quorum thresholds and voting periods are correctly implemented and equally enforced for all participants (Shah, 2024).
4. **Transparency and Auditability:** Smart contracts enable transparent governance by recording all actions on-chain. Evaluation includes verifying whether voting data and governance decisions are accessible, verifiable, and reproducible from the blockchain (Messias et al., 2024).
5. **Efficiency and Cost:** The efficiency of voting mechanisms is often limited by gas costs. On-chain voting increases security but may reduce participation due to high fees. Benchmarking compares the gas consumption of different mechanisms and considers trade-offs in hybrid models (e.g., off-chain voting with on-chain execution) (Messias et al., 2024).
6. **Governance Rule Implementation:** Contracts must correctly implement rules such as proposal submission, voting logic, quorum checks, and execution (Shah, 2024). Evaluation involves reviewing the alignment between governance specifications and their smart contract implementation.

7. **Upgradability:** Governance systems must support future changes. Evaluation includes whether the smart contracts are upgradable via governance processes or proxy patterns, and whether such mechanisms maintain decentralization and security (Eisermann et al., 2025).

In summary, although standardized benchmarks for evaluating blockchain governance systems remain limited in existing literature, these common criteria provide a useful framework for assessing the performance, security, and fairness of voting mechanisms. In the context of multi-chain governance systems, these evaluation dimensions are particularly important to ensure secure, scalable, and reliable cross-chain proposal management and voting.

2.7. Chapter summary

This chapter reviewed existing literature on blockchain governance, with a focus on voting mechanisms in both single-chain and multi-chain environments. It discussed the limitations of traditional governance systems, the emergence of DAOs, and various voting models. Technologies including smart contracts, cross-chain communication technologies, and IPFS were examined for their roles in enabling decentralized governance. Finally, key evaluation criteria such as security, fairness, transparency, and cost-efficiency were presented to guide the assessment of governance systems, especially in multi-chain settings.

CHAPTER 03: METHODOLOGY

3.1. Chapter overview

This chapter explains the research, design, and development methodology for building a multi-chain governance system focused on improving proposal management and voting. It discusses key challenges identified from the literature review and outlines the research methods and development processes to be followed. Furthermore, it describes the project management approach, resource requirements, and risk mitigation strategies necessary to ensure the success of the project.

3.2. Research methodology

Layer	Choice	Justification
Philosophy	Positivism	The study is grounded in objective, measurable outcomes such as automation efficiency, reduced deployment overhead, and secure cross-chain synchronization.
Approach	Deductive	The research tests and validates an architecture based on existing theories of smart contract reusability, automation, and blockchain interoperability.
Methodological Choice	Quantitative	The evaluation focuses on measurable factors such as automation success rate, reduction of manual steps, gas cost optimization, and security validation.
Strategy	Experimental	A multi-chain governance automation system will be implemented and tested on blockchain testnets through simulated proposal and voting scenarios.
Time Horizon	Cross-sectional	Data will be collected at a single point in time during system testing, rather than through longitudinal observations.
Techniques and procedures	Data collection, Literature Review	Data will be collected from system logs, smart contract deployments, testnet activities, and analysis of related literature.

Table 2: Research methodology

3.3. Development methodology

3.3.1. Requirement elicitation methodology

To gather the system requirements, following methods will be used,

- **Literature Review:** A comprehensive review of existing multi-chain governance frameworks, smart contract reusability techniques, and cross-chain interoperability methods will be conducted to identify limitations and best practices.
- **Surveys:** Structured surveys will be distributed among blockchain developers, DAO contributors, and relevant stakeholders to gather insights on challenges, expectations, and requirements for automated governance systems.

3.3.2. Design methodology

The system will be designed following the **Object-Oriented Analysis and Design Methodology (OOADM)**. This methodology was chosen due to its suitability for modular, component-based architecture, allowing individual features such as smart contract modules, relayer logic, and governance proposal handlers to be developed and tested independently.

3.3.3. Programming paradigm

The development will use **Object-Oriented Programming (OOP)** for both the smart contract structure and relayer to ensure modularity, code reusability, and easier maintenance as the system scales.

3.3.4. Testing methodology

A structured testing methodology will be adopted, including the following levels:

- **Unit Testing:** Individual smart contract functions and relayer modules will be tested using blockchain development tools such as Hardhat to validate core logic.
- **Integration Testing:** The interaction between smart contracts and the relayer will be tested across blockchain testnets to verify automated proposal deployment, cross-chain synchronization, and voting aggregation processes.
- **Security Testing:** Both manual and automated testing techniques will be employed to identify vulnerabilities, including replay attacks, unauthorized proposal manipulation, and cross-chain communication failures.

3.4. Project management methodology

This project will use the **Agile project management methodology**, as it supports flexibility and iterative improvement. Agile enables the project to be divided into smaller phases with regular planning, testing, and feedback. This is especially important for smart contract development, where deployed contracts cannot be changed and must be redeployed if errors are found.

3.4.1. Schedule

3.4.1.1.Gantt chart

Gantt Chart is available in **Appendix B: Gantt Chart**

3.4.1.2.Deliverables and dates

Deliverable	Date
Literature review	11th Nov 2024
Software Requirement Specification	
Proof of Concept	
Design Document	2nd Feb 2025
Prototype	
Interim Project Demo	
Implementation	7th Apr 2025
Testing	
Evaluation	
Thesis Submission	
Minimum Viable Product	

Table 3: Deliverables and Dates

3.5. Resources

3.5.1. Hardware resources

- **Apple M1 Chip:** To handle cross-chain interoperability development, basic simulations, and manage development tools.
- **8GB Unified Memory or More:** To support development tasks and testing environments, allowing smooth multitasking.

- **50GB SSD or More:** To store the application code, development tools, and project files and other necessary documentations.

If hardware resources are insufficient, a cloud-based environment (e.g., CodeSandbox, Stackblitz, Google Cloud, AWS) will be utilized for development.

3.5.2. Software resources

Category	Software	Purpose
Operating System	MacOS	Development and testing environments
Development Tools	VS Code, Remix	Code editing and debugging
Smart Contract Tools	Hardhat	Smart contract development and testing
Programming Languages	JavaScript, TypeScript, Solidity	System and smart contract development
Frameworks and Libraries	React.js, Node.js, Web3.js	Frontend and backend development
Version Control	Git, GitHub	Code versioning
Testing Tools	Chai	Automated testing frameworks and security testing tools
Libraries	OpenZeppelin	Implementing battle-tested reusable smart contracts
Documentation Tools	Microsoft Word	To create the project documents

Table 4: Software Requirements

3.5.3. Technical skills

- **Blockchain Fundamentals:** Understanding blockchain basics, governance model expertise and security principles.
- **Smart Contract Development:** Knowledge in Solidity and secured smart contract coding practices.
- **Web Development:** Frontend and backend skills in JavaScript, React.js, and Node.js for developing UI and other services.
- **Security Practices:** Knowledge in securing smart contracts, protecting against common vulnerabilities, and maintaining data privacy in cross-chain communication.

3.6. Risks and mitigation

Risk	Severity	Frequency	Mitigation Strategy
Automation logic failing across chains	High	Likely	Design fallback mechanisms and logging within the relayer to detect and recover from sync failures.
Smart contract bugs or security vulnerabilities	High	Likely	Perform unit testing, code reviews, and use established libraries for critical modules.
Time constraints impacting implementation	Medium	Likely	Prioritize core features using Agile sprints and monitor progress with regular checkpoints.
Tooling or deployment issues on testnets	Medium	Frequent	Use stable, well-documented testnets and prepare alternative chains
High gas costs or network congestion	Medium	Rare	Use testnets with faucet support and optimize contract logic for minimal gas usage during testing.
Relayer failure or event listener disconnect	High	Likely	Implement WebSocket reconnect logic and periodic polling as backup to avoid data loss.

Table 5: Associated Risks and Mitigations

3.7. Chapter summary

This chapter outlined the research and development methodologies used in the project, including the research approach, requirement elicitation methods, design strategy, programming paradigm, and testing approach. It also covered the project management methodology and identified key risks with corresponding mitigation strategies. These methodologies provide a structured foundation for the design, development, and evaluation of the automated multi-chain governance system.

CHAPTER 04: SOFTWARE REQUIREMENTS SPECIFICATION

4.1. Chapter overview

This chapter outlines the process of identifying the software requirements for the proposed system. It begins with a rich picture to illustrate the problem domain and includes an analysis of key stakeholders. Several requirement elicitation methods are used to gather relevant information, which helps define the system's functional and non-functional requirements. Furthermore, diagrams such as the context diagram and use case diagram are used to present the system's scope and interactions.

4.2. Rich picture

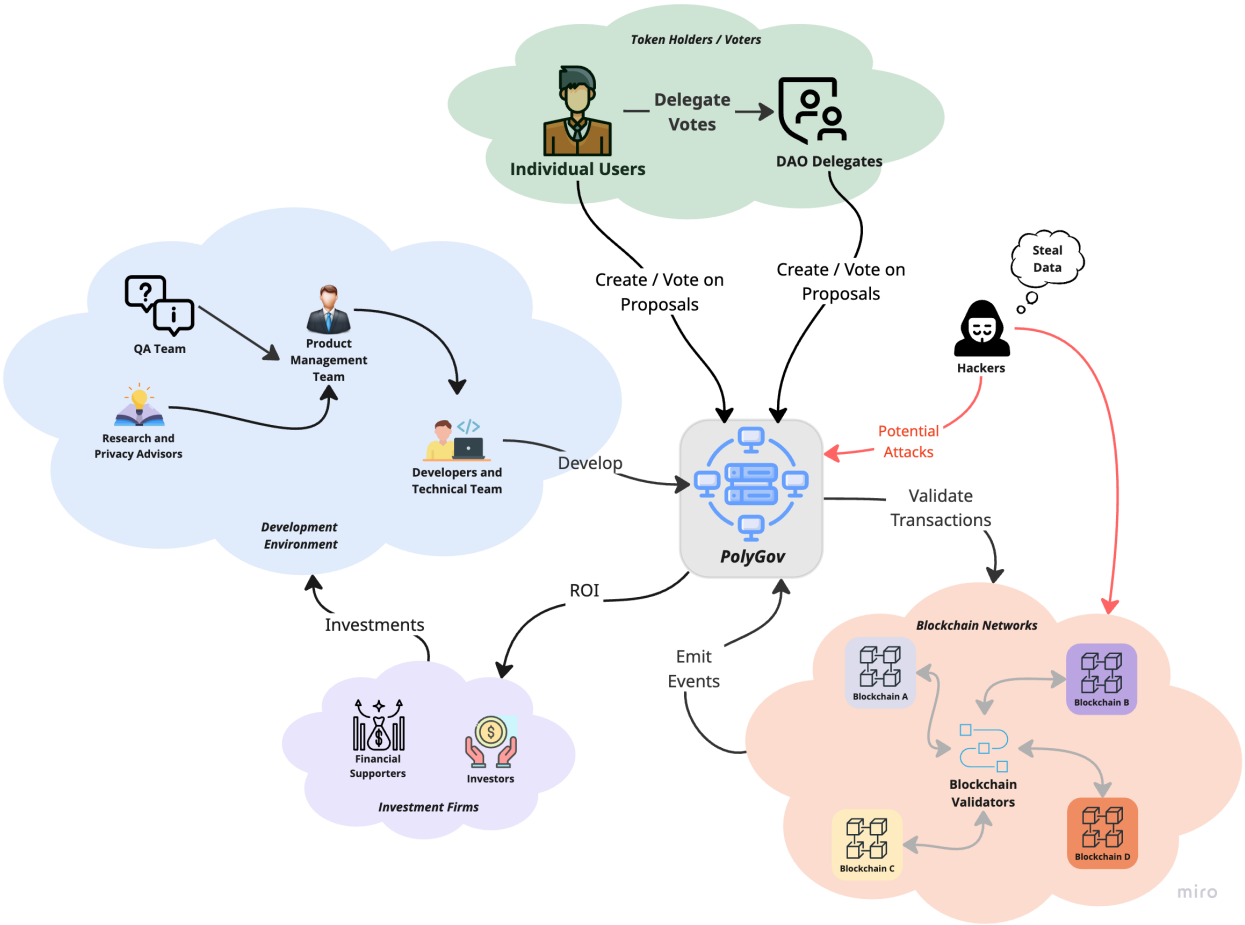


Figure 4: Rich picture diagram (self-composed)

4.3. Stakeholder analysis

4.3.1. Stakeholder onion model

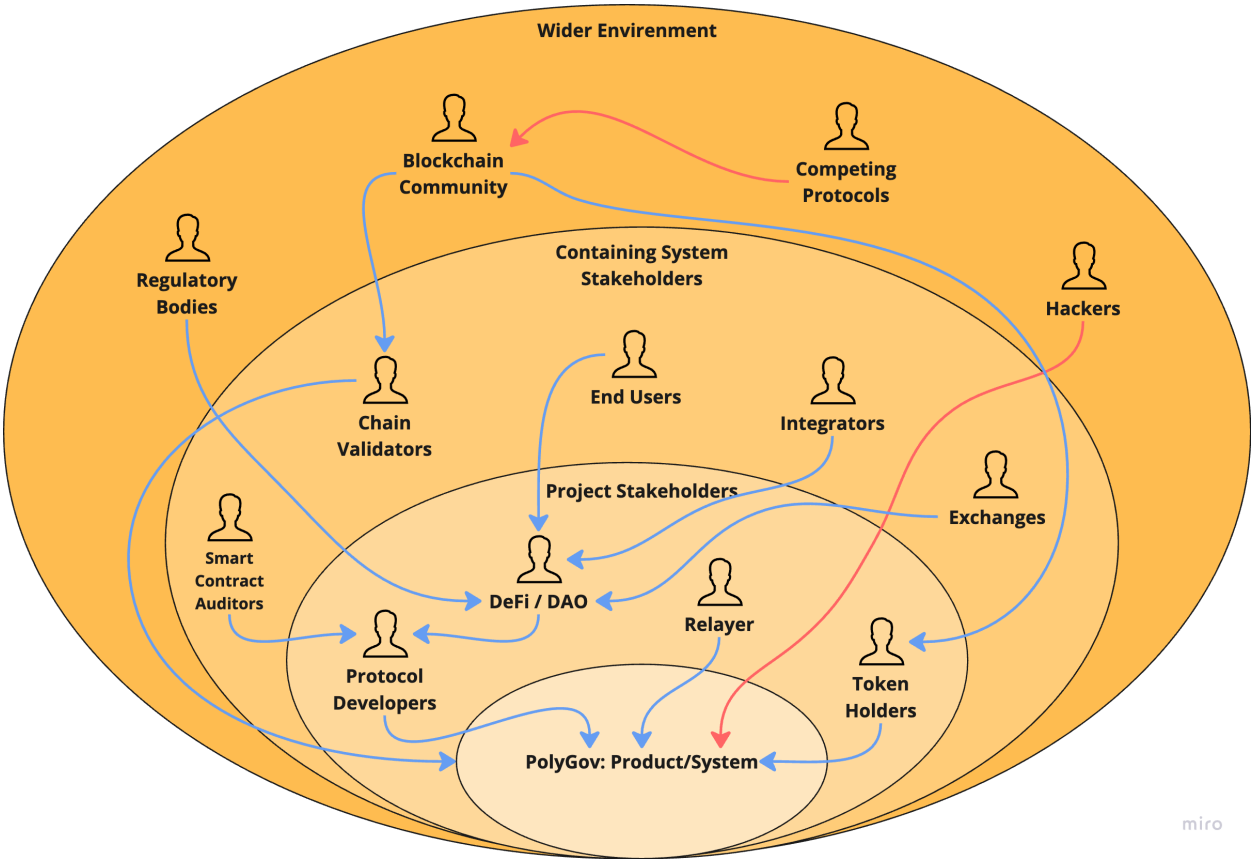


Figure 5: Stakeholder onion model (self-composed)

4.3.2. Stakeholder viewpoints

Stakeholder	Role	Description
The Project Stakeholders		
Token Holder / User	Functional Beneficiary	Token holders propose and vote on governance changes. They want a user-friendly, transparent system where their votes are correctly counted across all blockchains.
Relayer	Infrastructure Operator	Handles cross-chain operations by mirroring proposals and aggregate votes. Focused on keeping the system reliable, fast, and in sync.

Protocol Developers	Technical Contributor	Build and maintain PolyGov's features. Aim to improve functionality, scalability, and add support for new chains to keep the system competitive.
DeFi/DAO	Functional Beneficiary	DeFi protocols and DAOs using PolyGov for governance. They want an efficient and reliable system to manage decisions fairly across multiple blockchains.
Containing System Stakeholders		
Auditor	Advisor/Regulator	Performs security checks on PolyGov. Ensures there are no vulnerabilities and that governance processes are secure and transparent.
Chain Validators	Infrastructure Operator	Validate blockchain transactions where PolyGov operates. They expect PolyGov to create valid transactions without causing network issues.
End Users	Indirect Beneficiary	Users who experience PolyGov through DeFi/DAO apps. They expect governance changes to happen smoothly without affecting their app experience.
Integrators	Technical Partner	Projects like wallets or dApps that connect to PolyGov. They need clear APIs and good documentation to integrate governance features easily.
Wider Environment Stakeholders		
Blockchain(s)	Infrastructure	The blockchains that store PolyGov's proposals and votes. They expect PolyGov to be efficient and not overload their networks.
Attacker	Negative Stakeholder	Malicious actors trying to exploit PolyGov. The system needs strong defenses against attacks like vote manipulation or fund theft.
Blockchain Community	External Influencer	The wider crypto ecosystem watching PolyGov. They care about how well it promotes decentralization, transparency, and governance standards.

Competing Protocols	Negative Stakeholder	Other governance systems competing with PolyGov. They push PolyGov to stay innovative and improve to maintain its market position.
Regulatory Bodies	External Regulator	Governments and regulators monitoring blockchain systems. They expect PolyGov to comply with legal requirements like anti-money laundering (AML) and privacy laws.

Table 6: Stakeholder viewpoints

4.4. Selection of requirement elicitation methodologies

To gather comprehensive and accurate requirements for the development of the multi-chain governance system, two methods were used: literature review and surveys. These methods were chosen to capture both theoretical insights and practical feedback from stakeholders in decentralized governance.

Literature Review:

- The literature review focused on analysing previous research related to blockchain governance, decentralized decision-making, and multi-chain systems.
- It provided insights into key challenges, limitations of existing systems, and potential design improvements to enhance governance functionality.

Surveys:

- A survey was conducted with key stakeholders, including token holders, developers and validators, to gather their preferences and feedback on governance features.
- The survey ensured that the system design reflects real-world needs, such as token threshold preferences, voting mechanisms, and cross-chain proposal synchronization.

4.5. Findings

4.5.1. Literature review findings

Finding	Citation
Traditional governance models are centralized, rigid, and lack transparency, making them unsuitable for decentralized environments.	Hamid Ekal and Abdul-wahab, 2022; Liu et al., 2023; Shah, 2024
Transparent, community-driven governance with open voting models and verifiable results improves user trust and engagement.	Yadav et al., 2024
High gas fees reduce voter turnout, especially among small token holders.	Feichtinger et al., 2023; Eisermann et al., 2025
Single-chain governance lacks interoperability, limiting multi-chain participation.	Frangella and Herskind, 2022; Ao et al., 2023
Smart contracts automate governance rules but are challenging to benchmark due to cost, complexity, and upgrade limitations.	Liu et al., 2023; Shah, 2024; Messias et al., 2024
Existing systems like MULTAV introduces cross-chain voting but suffers from deployment overhead and scalability issues.	Fan, Chai and Zhong, 2020
Relay-based cross-chain systems offer a trust-minimized and modular approach for governance proposal and vote synchronization.	Ou et al., 2022; Sion et al., 2024
IPFS can store governance metadata off-chain, improving performance and cross-chain data access.	Belchior et al., 2021; Sion et al., 2024

Table 7: Literature review findings

4.5.2. Survey findings

A structured questionnaire containing both multiple-choice and open-ended questions was distributed to stakeholders to gather insights into their requirements for decentralized governance. The key survey area are preferences on proposal creation, voting methods, and cross-chain synchronization and feedback on token thresholds and decentralized relayer services.

Survey Summary:

- **Multi-Chain Governance**: Most participants emphasized the importance of supporting governance across multiple blockchains, with a preference for a primary chain model with secondary chain mirroring.
- **Voting Mechanisms**: Snapshot-based voting was considered essential to prevent token manipulation. Most respondents preferred a simple one-token-one-vote mechanism, though some support for quadratic and delegated voting was also observed.
- **Proposal Requirements**: Stakeholders supported a minimum token threshold for creating proposals to avoid spam, but many recommended that thresholds should be flexible or adjustable.
- **Relayer Trust**: Most participants found a trusted off-chain relayer acceptable for cross-chain synchronization, preferring simplicity while highlighting the need for strong security measures.
- **Security Concerns**: Governance manipulation (Sybil attacks and vote buying), smart contract vulnerabilities, and relayer compromises were identified as major risks that need mitigation.
- **Vote Aggregation and Synchronization**: There was a preference for decentralized, on-chain aggregation over centralized methods, and respondents expected cross-chain votes to be mirrored and finalized quickly.
- **User Interaction**: Most users preferred interacting through a web-based dApp with features like clear proposal listings, real-time updates, and wallet integrations.

Survey response analytics are available in **Appendix C: Survey Findings**

4.6. Summary of findings

Finding	Elicitation Method
Traditional governance models are not suitable for decentralized systems due to centralization, lack of transparency, and rigidity.	Literature Review
Multi-chain governance is essential, with preference for a primary chain and secondary chain mirroring model.	Literature Review, Survey

High gas fees and technical complexity reduce participation and inclusivity in governance processes.	Literature Review
Snapshot-based voting and one-token-one-vote models improve fairness and participation.	Literature Review, Survey
A minimum token threshold for proposal creation is necessary to prevent spam but should remain flexible.	Survey
Trusted off-chain relayer usage for cross-chain synchronization is acceptable if security measures are ensured.	Literature Review, Survey
Preference for decentralized, on-chain vote aggregation for improved transparency and trust.	Survey
IPFS can be used to store governance metadata off-chain to enhance scalability and data accessibility.	Literature Review
Users prefer a web-based dApp interface with real-time updates, clear proposal listings, and wallet integrations.	Survey

Table 8: Summary of Findings

4.7. Context diagram

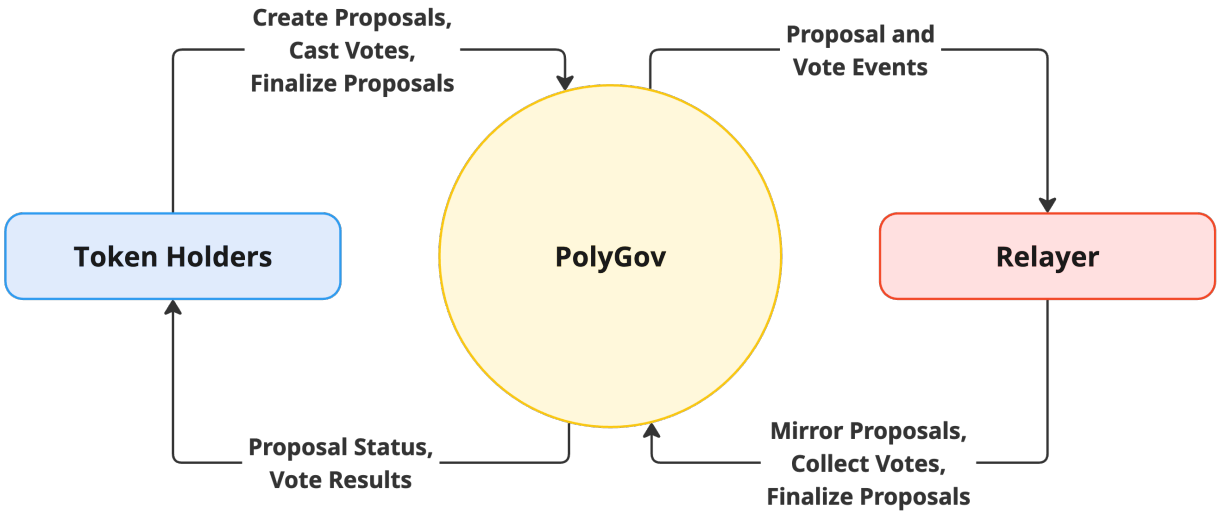


Figure 6: Context Diagram (self-composed)

4.8. Use case diagram

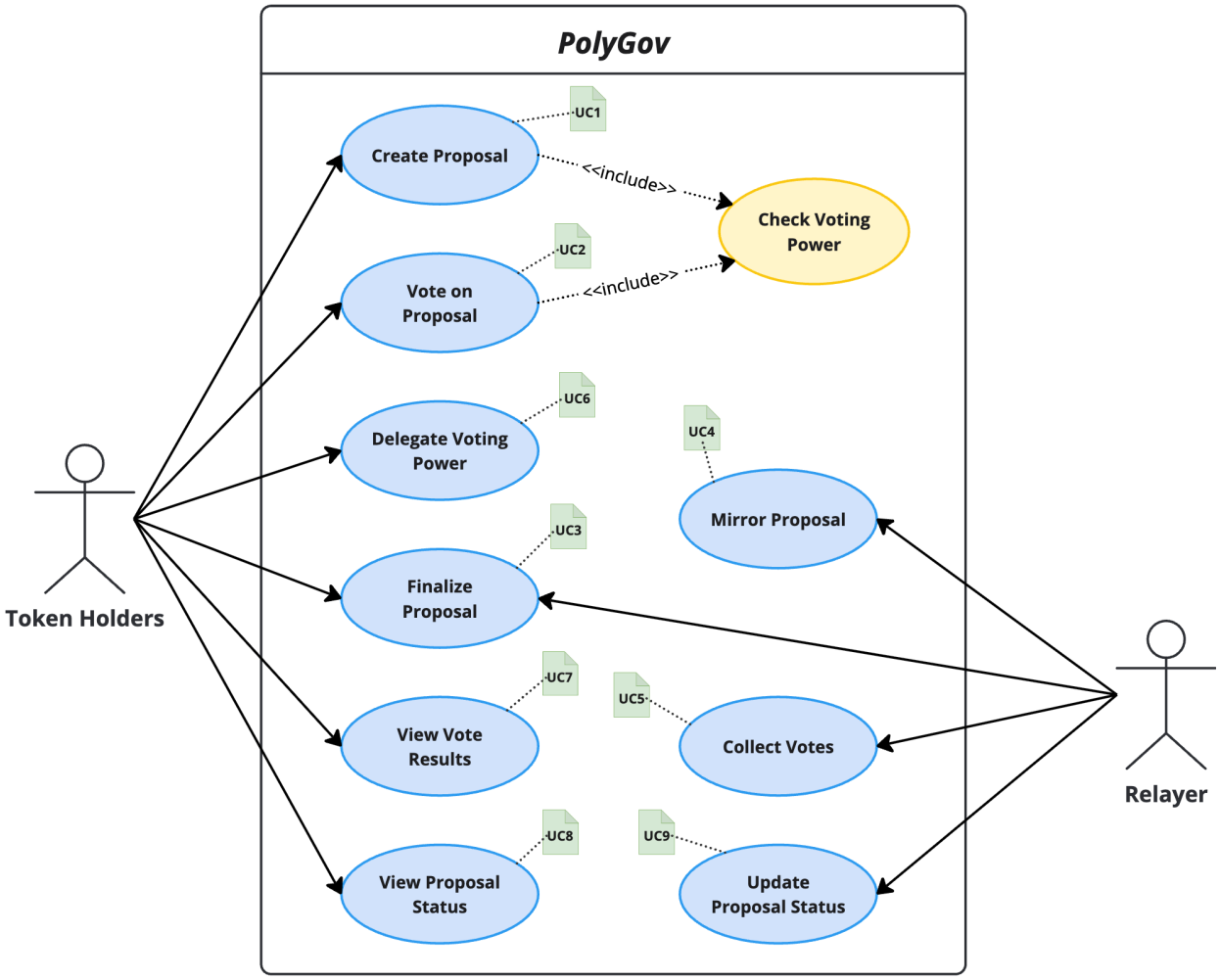


Figure 7: Use Case Diagram (self-composed)

4.9. Use case description

The primary use case descriptions are provided below. Additional use cases can be found in **Appendix D: Use Case Descriptions**.

Use case	Mirror Proposal
ID	UC4
Actors	Relayer
Description	The Relayer Service replicates a proposal from the main blockchain onto other secondary blockchains, enabling users on those chains to participate in the voting process.

Precondition	- A proposal exists on the main blockchain. - It has not been mirrored to the target blockchain. - The Relayer Service is connected to all relevant blockchains.
Main flow	1. The Relayer Service detects a new main blockchain proposal. 2. Proposal details are retrieved. 3. For each secondary blockchain, the Relayer checks if the proposal already exists. 4. If not exist, the proposal is copied and mirrored to the secondary blockchains. 5. The proposal is made available for voting.
Alternative flows	A1: Proposal Already Exists - At step 3, the system skips to the next chain if the proposal is already present A2: Mirroring Missed Proposals - The Relayer checks for older, uncopied proposals and mirrors and syncs them as needed.
Exceptions	- Connection issues prevent the proposal from being copied to certain blockchains. - Target blockchain systems are unavailable.
Postconditions	- The proposal appears on the other secondary blockchain with consistent details for voting. - Voting becomes possible on the mirrored blockchain.

Table 9: Use case description for proposal mirroring

4.10. Requirements

The ***MoSCoW*** method is used in here to prioritize the requirements for this proposed system

- M (Must Have) – Requirements necessary for the completion of project.
- S (Should Have) – Requirements that are important, but not essential.
- C (Could Have) – Nice to have requirements.
- W (Won't have) – Requirements that are not important.

4.10.1. Functional requirements

ID	Requirement	Priority
FR01	Users must be able to create governance proposals on the main blockchain, specifying title, description, voting period, and executable actions.	M
FR02	Users must be able to vote on active proposals on both main and secondary blockchains, weighted by their voting power.	M
FR03	The system must finalize proposals after voting ends, aggregating votes from the main and secondary chains securely.	M
FR04	The relayer must mirror proposals from the main chain to secondary chains automatically via events.	M
FR05	The relayer must collect finalized votes from secondary chains and submit them to the main chain for aggregation.	M
FR06	The system should allow users to view proposal details, status (pending, accepted, rejected, executed), and vote tallies.	S
FR07	The system could execute accepted proposals by performing predefined external contract actions.	C
FR08	The system should prevent unauthorized actions (e.g., non-relayer trying to mirror or finalize votes).	S
FR09	The system could allow users to delegate voting power to others (proxy voting).	C
FR10	The system could integrate with external systems (oracles, off-chain data feeds) to enhance governance decisions.	W

Table 10: Functional Requirements

4.10.2. Non-Functional requirements

ID	Requirement	Priority
NFR01	The system must ensure data consistency across the main and secondary chains, keeping proposals and votes synchronized.	M
NFR02	The system must protect against unauthorized access, ensuring that only eligible users can create proposals or cast votes.	M

NFR03	The system must be resilient to network disruptions (e.g., chain downtime or fork events) without compromising the voting process.	M
NFR04	The system should provide a user-friendly interface for interacting with proposals, voting, and monitoring outcomes.	S
NFR05	The system should maintain an average response time of less than 5 seconds for user actions such as creating a proposal or voting.	S
NFR06	The system should be scalable to support additional secondary chains without major changes to the architecture.	S
NFR07	The system could provide real-time alerts and notifications for major governance events (e.g., proposal created, voting ended).	C
NFR08	The system support analytics and reporting features (e.g., voter turnout, proposal success rate).	W

Table 11: Non-Functional Requirements

4.11. Chapter summary

This chapter describes the software requirements for the proposed multi-chain governance voting system. It includes a rich picture to explain the problem and an analysis of key stakeholders. Requirements were gathered through literature reviews and surveys, highlighting the need for the system. The chapter also presents a context diagram, use case diagram, and the use case descriptions. Finally, it lists and prioritizes the functional and non-functional requirements using the MoSCoW method.

CHAPTER 05: SOCIAL, LEGAL, ETHICAL AND PROFESSIONAL ISSUES (SLEP)

5.1. Chapter overview

This chapter discusses the social, legal, ethical, and professional (SLEP) issues related to the proposed system. It highlights the importance of addressing these concerns during system development to ensure responsible and lawful practices. The chapter identifies potential SLEP issues and outlines appropriate strategies to mitigate them. By considering these factors, the system aims to meet legal requirements, respect user rights, and follow professional standards in software development.

5.2. SLEP issues and mitigation

5.2.1. Social considerations

The system was designed to be neutral, with no influence from religious, political, or cultural biases. Its main purpose is to support fair and transparent decision-making in decentralized settings. Surveys and interviews were conducted with relevant stakeholders to understand their governance needs. All participants were informed about the study's purpose and how their feedback would be used. The user interface was developed to be inclusive and accessible for users across various blockchain platforms.

5.2.2. Legal considerations

The system does not store any personal or sensitive data, helping to ensure privacy and legal compliance. All smart contracts created for the project are source code verified and publicly available on their respective blockchains and are licensed under the MIT open-source license, supporting transparency and external review.

Academic materials used during the research were accessed through the university's institutional access. In cases where content was restricted, full-text copies were obtained directly from the authors with their permission. Evidence of these permissions is provided in Appendix E: Full-Text Paper Request**.**

5.2.3. Ethical considerations

All participants gave informed consent before taking part in the study. Their roles and contributions were clearly explained. Ethical concerns such as vote manipulation, Sybil attacks, and the risk of compromised relayers were considered in the system design. Measures were put in place to reduce these risks and ensure fairness, trust, and accountability within the system.

5.2.4. Professional considerations

This work follows accepted academic and professional standards. All external sources, images, and third-party content are properly cited. Large Language Models (LLMs) were used to assist throughout the research process in accordance with university guidelines. All content was reviewed by the author to ensure academic integrity.

The full source code for the smart contracts is published under the MIT license to encourage openness, collaboration, and further development. Secure coding practices were followed to minimize bugs and security issues in the smart contracts. The system's known limitations are discussed in the evaluation to provide a fair and transparent assessment.

5.3. Chapter summary

This chapter reviewed the key social, legal, ethical, and professional responsibilities involved in the development of a decentralized governance system. Actions such as open-source licensing, informed consent, and secure, ethical design were taken to ensure the project met recognized research and development standards.

CHAPTER 06: DESIGN

6.1. Chapter overview

This chapter describes the design of the multi-chain governance voting system, including its architecture, components, workflows, and user interfaces. The system is designed to support decentralized proposal creation, voting, and execution across multiple blockchains while maintaining synchronization and consistency. The chapter explains how the design goals are achieved and includes diagrams illustrating the system's architecture, proposal lifecycle, and data flow to provide a clear understanding of the overall design.

6.2. Design goals

Design Goal	Description
Interoperability	The system must synchronize proposals and votes across multiple blockchain networks using unique identifiers, ensuring a seamless governance process across chains.
Security	As blockchain governance systems are prime targets for attacks such as vote manipulation, the system must protect against unauthorized access and tampering, maintaining the integrity and trustworthiness of the governance process.
Transparency	All proposals, votes, and outcomes must be fully visible and auditable across all participating blockchains to build trust and accountability among users.
Scalability	The system must easily accommodate additional blockchains with minimal configuration changes, enabling the governance network to grow without redesigning core components.
Usability	A user-friendly interface must be provided for creating proposals, voting, and viewing results, minimizing the steps required to participate in governance tasks and enhancing overall user engagement.

Table 12: Design Goals of the proposed system

6.3. System architecture design

6.3.1. System architecture diagram

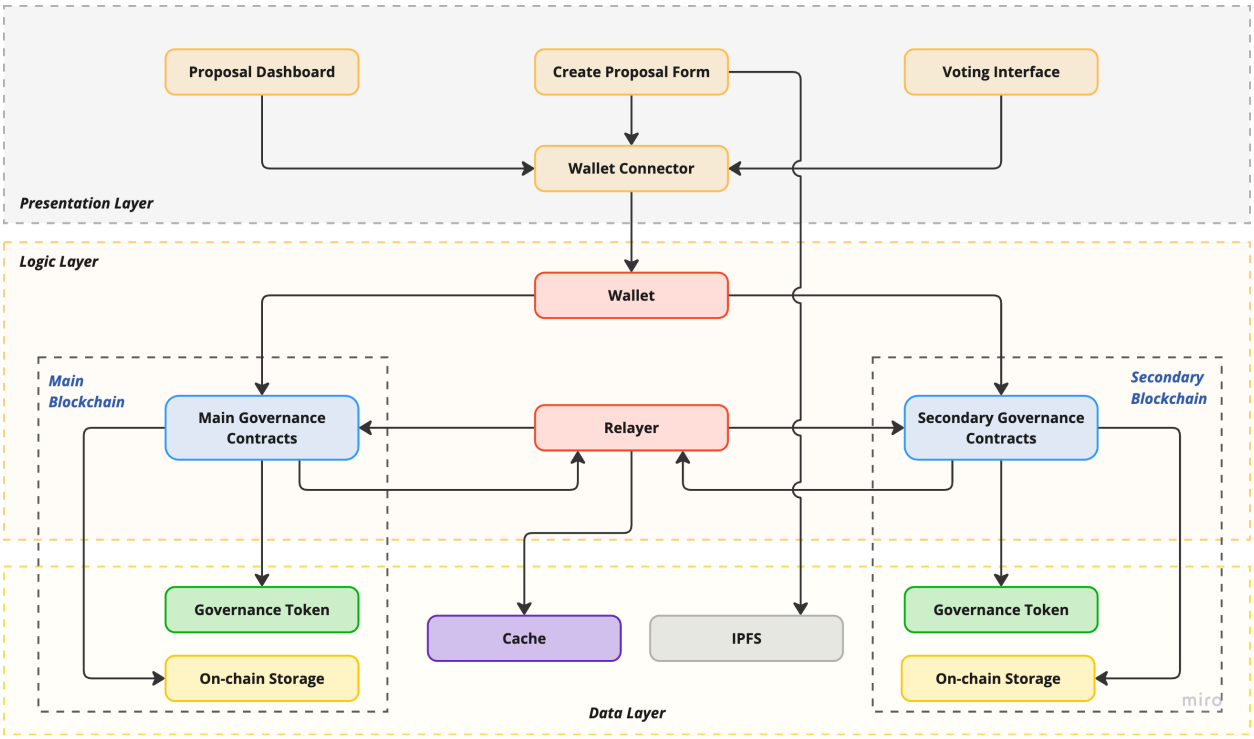


Figure 8: System architecture diagram (self-composed)

6.3.2. Discussion of system architecture layers

Presentation Layer

The Presentation Layer represents the user interface (UI) where participants interact with the governance system. It provides an intuitive and accessible environment for interacting with decentralized protocols without requiring technical blockchain expertise.

Key roles of the Presentation Layer:

- **Proposal Creation:** Users can initiate new governance proposals using the Create Proposal Form. Inputs such as the proposal title and voting duration are submitted through this interface. Additional metadata is uploaded to IPFS during this process, ensuring decentralized storage of proposal details.
- **Vote Casting:** Through the Voting Interface, users can cast “yes” or “no” votes. The wallet integration displays the user's token balance and voting power, ensuring transparent participation.

- **Proposal Tracking:** The Proposal Dashboard allows users to view live and historical proposals, check current voting tallies, and track the outcome of finalized proposals.

This layer integrates with wallet connection libraries, enabling transaction signing and blockchain interaction from within the web application.

Logic Layer

The Logic Layer encapsulates all application logic and coordination mechanisms, both on-chain and off-chain. It includes smart contracts deployed across multiple blockchains and a relayer service that facilitates cross-chain synchronization.

Key components of the Logic Layer:

- **Governance Smart Contracts:** Deployed separately on the main chain and secondary chains, these contracts manage proposal lifecycle, voting, and status updates.
- **Relayer:** The relayer acts as a bridge between blockchains. It monitors events such as proposal creation and voting, ensuring proposals are mirrored and votes are aggregated across chains.

Key roles of the Logic Layer:

- **Proposal Lifecycle Management:** Handles creation, state transitions (e.g., active, ended, finalized), and execution of governance proposals.
- **Voting Power Validation:** Enforces snapshot-based voting power using ERC20Votes, ensuring integrity by referencing token balances at proposal start time.
- **Cross-Chain Proposal Mirroring:** Proposals created on the main chain are automatically mirrored to secondary chains by the relayer.
- **Vote Collection and Aggregation:** After the voting period ends, the relayer fetches tallied votes from secondary chains, aggregates them, and updates the proposal status on all chains accordingly.

Data Layer

The Data Layer includes all persistent components where state is stored, both on-chain and off-chain components. This layer ensures immutability, transparency, and tamper-resistance of governance data.

Key roles of the Data Layer:

- **Proposal Storage:** All proposals and their metadata (e.g., title, status, voting times) are stored immutably on the respective blockchain.
- **Token Balances and Voting Power:** Token balances and delegated voting power are stored using ERC20Votes and are used to compute vote weights at the time of proposal creation.
- **Vote Storage:** Raw vote counts (yes/no) are recorded on each chain where votes are cast, until they are collected and finalized.
- **Final Result Storage:** After aggregation by the relayer, the final vote tallies and decision status are written back to the main chain and propagated to secondary chains.
- **Relayer Cache:** The off-chain relayer maintains a local cache (JSON-based) to track processed proposals and prevent redundant processing. This enhances efficiency and ensures consistency during synchronization.
- **IPFS:** IPFS is used to store proposal metadata (such as detailed descriptions and supporting documents) in a decentralized and tamper-proof manner. Smart contracts reference these IPFS hashes to maintain data integrity while minimizing on-chain storage costs.

6.4. Detailed design

6.4.1. Selection of design paradigm

The goal of this project is to design and implement a decentralized, multi-chain governance proposal and voting management system that involves complex components such as smart contracts, cross-chain relayer services, and modular user interfaces. These components must interact asynchronously and maintain a high level of independence and modularity. Among OOAD and SSADM, **Object-Oriented Analysis and Design (OOAD)** is the more suitable approach for this project. OOAD supports modular and component-based design, making it easier to model real-world entities as individual objects with their own attributes and behaviours.

6.4.2. Class diagram

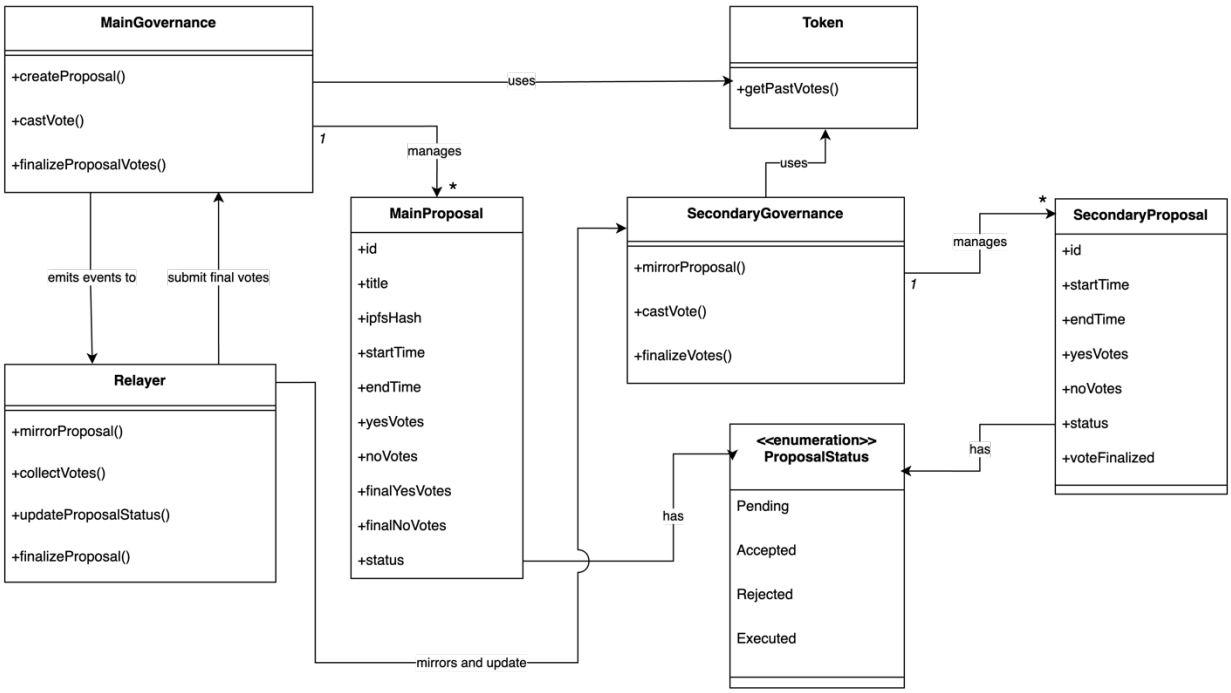


Figure 9: Class diagram (self-composed)

6.4.3. Smart contract design

The multi-chain governance system consists of three core smart contracts: the Main Governance Contract, the Secondary Governance Contract, and the Token Contract. Together, they manage proposal creation, voting, and secure cross-chain coordination, with the help of an off-chain relayer.

Token Contract

The Token Contract is based on the ERC20Votes standard, which supports snapshot-based voting. It determines voting power on each blockchain independently.

- **Voting Power:** A user's voting power is tied to the number of tokens they hold, captured at a specific block timestamp using getPastVotes. This prevents manipulation during voting.
- **Proposal Eligibility:** A minimum token balance (100 Tokens) is required to create proposals, ensuring only users with a stake in the system can initiate changes.
- **Chain-specific Deployment:** Each blockchain has its own token contract instance, supporting localized governance.

Main Governance Contract

The Main Governance Contract manages proposals and vote aggregation across all chains. It is deployed on the main blockchain and is responsible for:

- **Proposal Creation:** Users can create proposals by submitting a title, description, and duration. Each proposal is assigned a unique ID (via hash) and stored on-chain.
- **Vote Casting:** Users vote using their snapshot voting power. Each vote is recorded and tallied on the main chain.
- **Vote Finalization:** After voting ends, the main contract collects results from secondary chains and finalizes the total tally.

Secondary Governance Contract

The Secondary Governance Contract is deployed on each secondary blockchain and is responsible for handling local voting:

- **Mirrored Proposals:** Proposals are mirrored from the main chain by the relayer, maintaining consistency across all chains.
- **Voting:** Users can cast votes on mirrored proposals. Votes are stored locally and finalized after the voting window closes.
- **Status Updates:** Once the main chain finalizes the proposal, the relayer updates the status on each secondary chain.

6.4.4. Relayer design

The relayer is an off-chain service responsible for cross-chain synchronization in the PolyGov system. It operates in two main phases:

1) Proposal Mirroring Phase, where it listens to the main chain for new proposals and mirrors them onto secondary chains.
2) Vote Collection and Finalization Phase, where it aggregates votes from secondary chains and finalizes the outcome on the main chain.

The relayer ensures seamless synchronization, reducing manual intervention and risks associated with cross-chain governance.

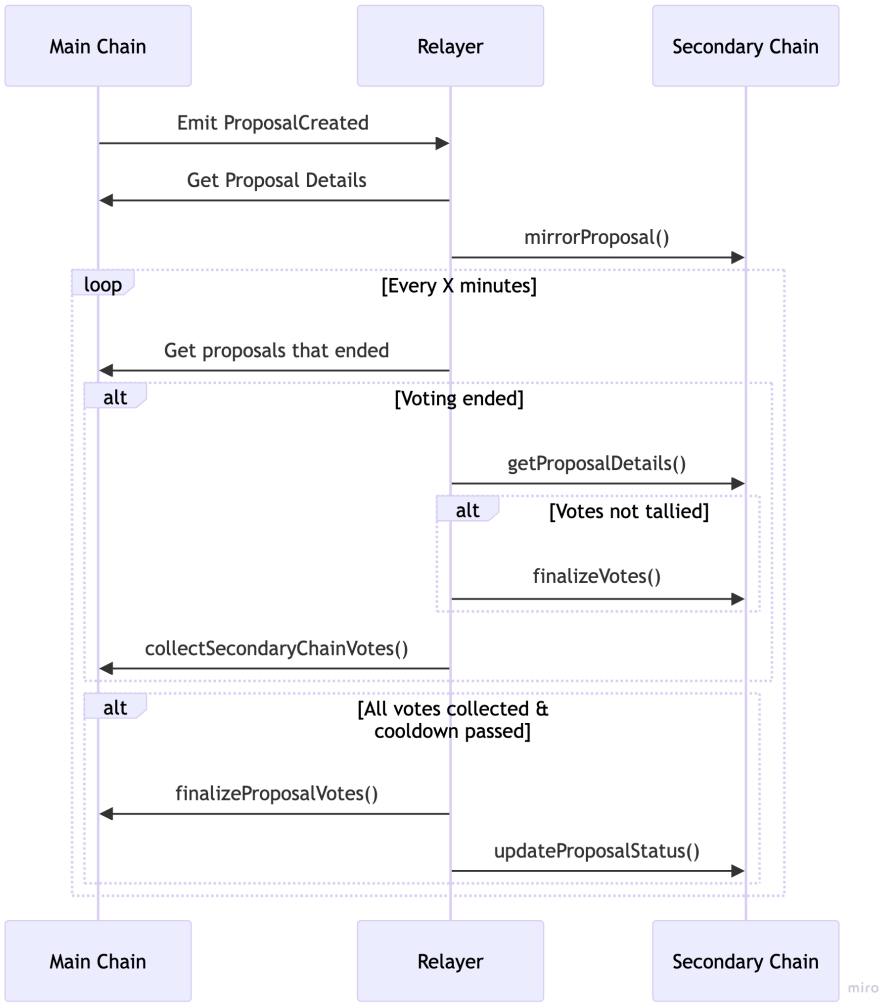


Figure 10: Relayer's sequence diagram

The diagram above illustrates the core lifecycle of a proposal across chains, showcasing the relayer's responsibility and coordination.

6.4.5. Vote Aggregation

After the voting periods conclude independently on both the main chain and the secondary chains, the system transitions into the Vote Collection and Finalization Phase. In this phase, the off-chain Relayer is responsible for collecting the finalized vote tallies from all secondary chains and submitting them to the main chain. Once all votes are collected, the main chain aggregates the votes from all participating chains, applies quorum validation, and finalizes the proposal outcome.

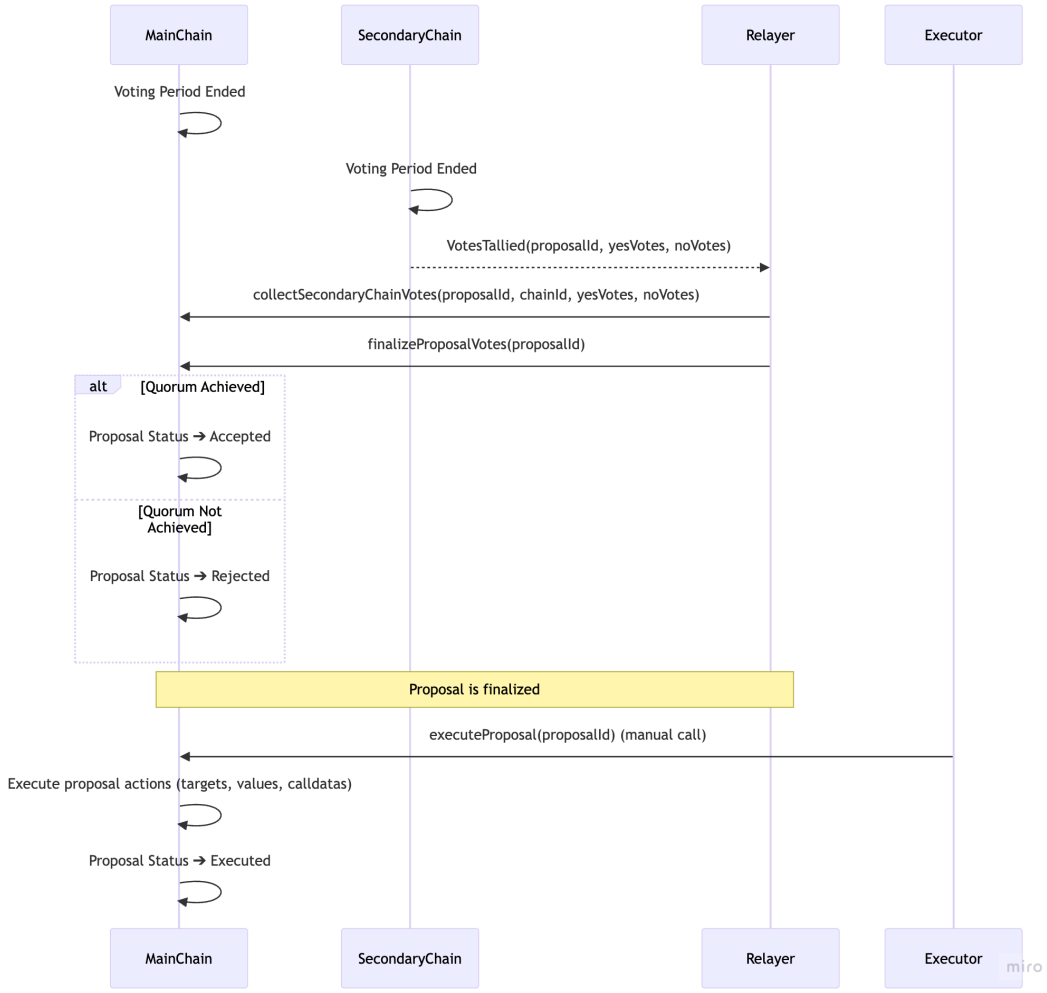


Figure 11: Vote aggregation sequence diagram

This design ensures that decentralized voting conducted across multiple blockchains is securely synchronized and aggregated into a single, tamper-resistant final result, maintaining the integrity and consistency of the governance process.

6.4.6. UI design

The user interface is designed to provide a clear and seamless experience for users engaging with the governance system (refer to Appendix F: Low Fidelity UI Design). Its main features include the **Proposal Dashboard**, which displays active proposals with key details such as titles, descriptions, voting deadlines, and statuses (active, passed, or rejected). The **Create Proposal Form** allows users to submit new proposals, while the **Voting Interface** offers a straightforward, one-click voting process and displays users' token balances and eligible voting power.

6.4.7. System process workflow

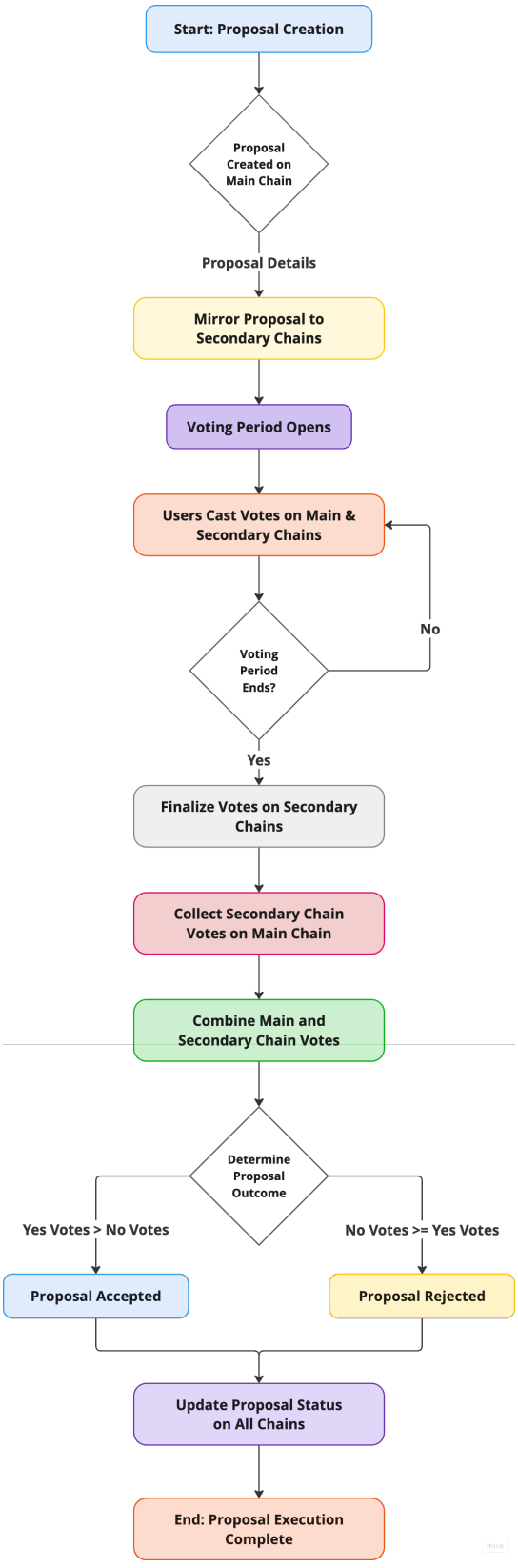


Figure 12: System workflow diagram (self-composed)

The Figure 12 and Figure 13 illustrates the full system workflow of the multi-chain governance process. It shows the complete lifecycle of a proposal. These visual workflow highlights how the system maintains synchronization and consistency between multiple blockchains during proposal management and voting activities.

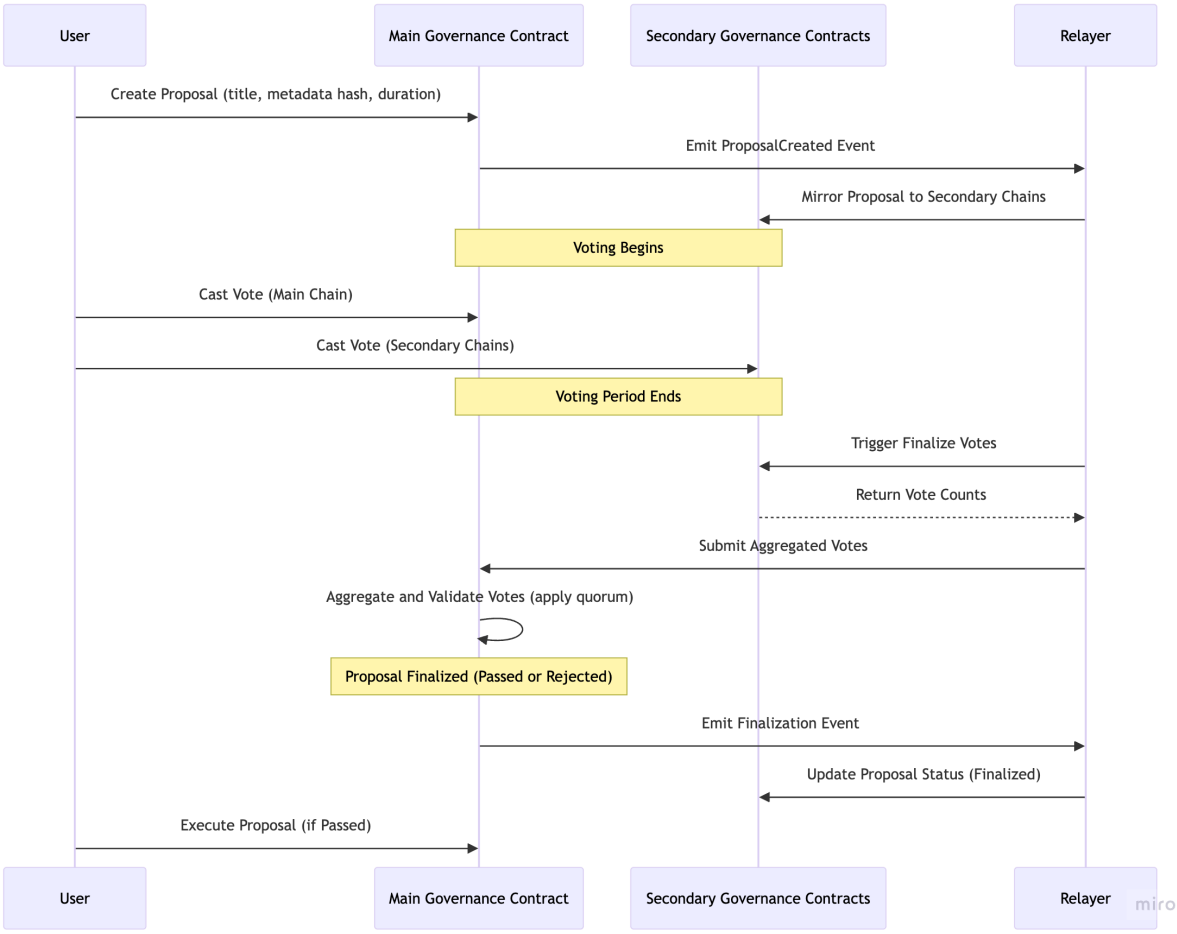


Figure 13: System workflow sequence diagram

6.5. Chapter summary

This chapter provided a comprehensive overview of the design of the multi-chain governance voting system, covering its architecture, core components, and workflows. The system is designed to maintain consistency in proposals and votes across multiple blockchains through the use of modular smart contracts and a trusted relayer mechanism. By focusing on scalability, security, and interoperability, the system facilitates decentralized decision-making across blockchains while ensuring a unified and coherent governance solution.

CHAPTER 07: IMPLEMENTATION

7.1. Chapter overview

This chapter explains the implementation of the multi-chain governance voting system, following the design specifications outlined in the previous chapter. It details the development of key components, including the smart contracts, relayer logic, and user interface. The chapter also describes the selected technology stack and demonstrates how the designed core functionalities were translated into a working solution, with examples and explanations.

7.2. Technology selection

7.2.1. Technology stack

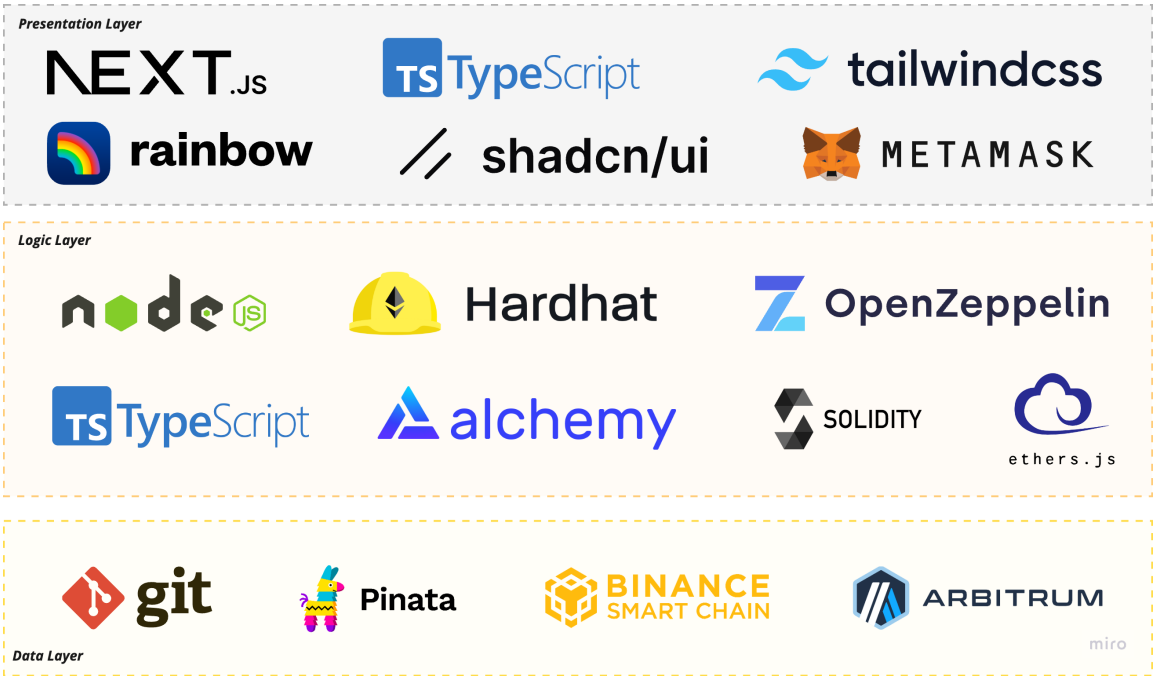


Figure 14: Technology stack (self-composed)

7.2.2. Programming language selection

- **Solidity:** Solidity was chosen for writing smart contracts due to its widespread use in EVM-compatible blockchains like Binance Smart Chain (BSC) and Arbitrum. It offers strong security features, maturity, and an extensive ecosystem of tools and libraries.
- **TypeScript:** TypeScript was used for the relayer and front-end development due to its static typing, which improves code maintainability and reduces runtime errors. It integrates seamlessly with Node.js and Next.js.

7.2.3. Development framework selection

- **Hardhat:** Hardhat is an advanced Ethereum development framework offering features like contract compilation, testing, and deployment. It was selected for its robust plugin system and ability to simulate local blockchains.
- **Next.js:** Next.js is a React-based full-stack framework with inbuild routes and optimizations, was used to build the front end, offering server-side rendering and static site generation for fast loading times and SEO optimization.

7.2.4. Libraries and tools selection

- **Ethers.js:** Ethers.js is a lightweight JavaScript library used to interact with the Ethereum blockchain. It simplifies key tasks such as sending transactions, querying on-chain data, and monitoring blockchain events.
- **RainbowKit (with wagmi):** RainbowKit provides a user-friendly interface for wallet connections and supports various wallets, including MetaMask. Combined with wagmi, it ensures seamless user interactions with the blockchain application.
- **OpenZeppelin:** OpenZeppelin libraries, known for their security and audits, were used to implement the ERC20 Token and governance-related mechanisms, ensuring secure and efficient smart contract development.
- **Pinata:** Pinata offers decentralized storage services, allowing the application to securely store files on the IPFS. It also provides a free usage tier, making it accessible for initial development and testing purposes.
- **Alchemy:** Alchemy provides reliable WebSocket and HTTP endpoints with support for multiple networks, offering speed, scalability, and real-time data access. This is especially useful for the relayer service to capture real-time blockchain events efficiently. Additionally, Alchemy offers a generous free tier, making it suitable for development and early-stage deployment.
- **shadcn/ui:** This library was utilized to create a customizable and reusable user interface, contributing to a clean and consistent design.
- **Tailwind CSS:** Tailwind CSS, a utility-first CSS framework, was chosen to speed up UI development and maintain design consistency across the application.

7.2.5. Blockchain networks and block explorers

- **BSC Testnet:** The Binance Smart Chain Testnet was selected for testing smart contracts in an EVM-compatible environment due to its fast transaction processing and low fees, making it ideal for development purposes.
- **Arbitrum Sepolia Testnet:** The Arbitrum Sepolia Testnet was used as an additional test network to demonstrate cross-chain functionality and ensure the system works efficiently with Layer 2 scaling solutions.
- **BSCScan Testnet Explorer**: BSCScan was used to monitor deployed contracts and on-chain transactions on the BSC Testnet.
- **Arbitrum Sepolia Explorer:** The Arbitrum Sepolia explorer was used to track and verify on-chain activities and deployments.

7.2.6. IDE selection

- **Remix IDE:** Remix was used during the initial development and testing of Solidity smart contracts due to its built-in feature such as testing environment, debugging tools, and immediate feedback capabilities.
- **VSCode:** VSCode was selected as the primary integrated development environment for the back-end (relayer) and front-end development. It offers extensive plugin support and built-in TypeScript capabilities, making it highly flexible and efficient.

7.2.7. Wallet and user interaction tools

- **MetaMask:** MetaMask was chosen as the primary wallet for users to create proposals and vote securely within the governance system.

7.2.8. Summary of technologies & tools selection

Component	Tools
Programming Languages	Solidity, Typescript
Development Frameworks	Hardhat, Next.js
Libraries	Ethers.js, RainbowKit, OpenZeppelin, shadcn/ui, Tailwind, Pinata, Alchemy
Blockchain networks	Arbitrum Sepolia, Binance Chain Testnet

IDEs	VSCode, Remix
Wallets	MetaMask
Version Control	Git, GitHub

Figure 15: Summary of technologies & tools selection

7.3. Implementation of core functionalities

7.3.1. Governance token

The **PGVToken** is an ERC20-based governance token that enables proposal creation and voting within the system. Users must hold a minimum token balance, as required by the governance contracts, to be eligible to create proposals. Each supported blockchain independently deploys its own instance of the PGV token contract, minting a total supply of 10,000 tokens to the deployer's address upon deployment.

```
contract PGVToken is ERC20, ERC20Permit, ERC20Votes {
    uint256 public constant MAX_SUPPLY = 10000 * 10 ** 18; // 10,000 tokens


    constructor() ERC20("PolyGov Token", "PGV") ERC20Permit(name()) {
        _mint(msg.sender, MAX_SUPPLY);
    }
```

Figure 16: Code - PGVToken Contract

```
// Override clock on snapshot from block number to block timestamp
function clock() public view override returns (uint48) {
    return uint48(block.timestamp);
}
```

Figure 17: Code - Overridden clock() function

The token contract is implemented in Solidity using OpenZeppelin's ERC20Votes and ERC20Permit extensions. It includes the following key features:

- **Timestamp-based vote snapshots:** Instead of block numbers, snapshots are based on block timestamps to enable consistent multi-chain governance.
- Auto-delegation to self if the user has not explicitly delegated voting power.

7.3.2. Proposal creation

The process of creating a proposal is managed by the Main Governance Contract. Users initiate proposals by providing a title, IPFS hash for additional details, voting duration, and optional execution parameters such as target addresses, ETH values, and call data. These proposals, if passed, can automatically execute on-chain actions.

To ensure eligibility, the contract verifies that the user holds sufficient voting power, requiring at least 100 PGV tokens at the time of proposal creation. Voting power is measured using snapshot-based logic to prevent last-minute token transfers from affecting voting outcomes.

Each proposal is assigned a unique bytes32 ID, generated using keccak256 over the proposer's address, current block number, and an incrementing proposal nonce. Proposal details, including the start and end times, voting status, and execution logic, are stored on-chain.

```
function createProposal(
    string memory _title, string memory _ipfsHash, uint256 _durationMinutes,
    address[] memory _targets, uint256[] memory _values, bytes[] memory _calldatas
) external {
    require(_durationMinutes > minVotingDuration, "Duration too short");
    require( _targets.length == _values.length && _targets.length == _calldatas.length,
        "Mismatched execution data");

    uint256 votingPower = governanceToken.getVotes(msg.sender);
    require(votingPower >= minCreationPower, "Insufficient voting power");

    proposalNonce++;
    bytes32 proposalId = keccak256(abi.encodePacked(msg.sender, block.number, proposalNonce));

    Proposal storage newProposal = proposals[proposalId];
    newProposal.id = proposalId;
    newProposal.title = _title;
    newProposal.ipfsHash = _ipfsHash;
    newProposal.startTime = block.timestamp;
    newProposal.endTime = block.timestamp + (_durationMinutes * 1 minutes);
    newProposal.proposer = msg.sender;
    newProposal.status = ProposalStatus.Pending;
    newProposal.targets = _targets;
    newProposal.values = _values;
    newProposal.calldatas = _calldatas;

    proposalIds.push(proposalId);
    emit ProposalCreated(proposalId, _title, newProposal.endTime);
}
```

Figure 18: Code - createProposal function on Main governance

Upon successful creation, a ProposalCreated event is emitted. This event is detected by the relayer, which mirrors the proposal across all registered secondary chains, ensuring cross-chain synchronization and enabling decentralized voting on multiple networks.

7.3.3. Proposal mirroring

The relayer plays a critical role in synchronizing proposals across all supported chains. Using a WebSocket connection and Ethers.js, the relayer continuously listens for the ProposalCreated event emitted by the Main Governance Contract on the main chain.

Upon detecting a new proposal, the relayer retrieves the proposal's metadata and invokes the mirrorProposal function on the Secondary Governance Contracts deployed across secondary chains. This function replicates the original proposal by storing essential details, including the title, IPFS hash, start time, end time, and proposer address.

```
contract.on("ProposalCreated", async (proposalId: string) => {
    console.log(`🆕 New proposal created: ${proposalId}`);
    try {
        const proposal = await mainGov.getProposalDetails(proposalId);
        for (const [chainId, { contract }] of Object.entries(secondaryConnections)) {
            const secondaryGov = new SecondaryGovernance(contract);
            await secondaryGov.mirrorProposal(
                proposalId,
                proposal.title,
                proposal.ipfsHash,
                proposal.startTime,
                proposal.endTime,
                proposal.proposer
            );
```

Figure 19: Code - Relayer - Listening to ProposalCreated Event

To prevent duplication, each secondary contract verifies whether a proposal with the same ID already exists before mirroring. Furthermore, only the trusted relayer is authorized to execute the mirroring process, enforced by the onlyRelayer access control mechanism, ensuring secure and controlled propagation of proposals.

```
function mirrorProposal(
    bytes32 proposalId, string memory _title, string memory _ipfsHash,
    uint256 _startTime, uint256 _endTime, address _proposer
) external onlyRelayer {
    if (proposals[proposalId].startTime > 0)
        revert ProposalAlreadyExists(proposalId);

    proposals[proposalId] = Proposal({
        id: proposalId,
        title: _title,
        ipfsHash: _ipfsHash,
        yesVotes: 0,
        noVotes: 0,
        startTime: _startTime,
        endTime: _endTime,
        proposer: _proposer,
        status: ProposalStatus.Pending,
        voteFinalized: false
    });

    proposalIds.push(proposalId);
    emit ProposalMirrored(proposalId, _title, _endTime);
}
```

Figure 20: Code - mirrorProposal function on SecondaryGovernance

This mirroring mechanism guarantees that all participating chains remain synchronized. As a result, users on any supported blockchain can view and vote on active proposals, maintaining a consistent and decentralized governance experience across the ecosystem.

7.3.4. Voting

Users can participate in the voting process on any supported chain by interacting with the governance contract's castVote function. The contract enforces three conditions to ensure a secure and fair voting mechanism:

1. **Proposal Validation:** The proposal must exist, and the voting period must still be active.
2. **One Vote Per User:** The user must not have already voted on the proposal, tracked through the hasVoted mapping.
3. **Snapshot-Based Voting Power:** The user must have held sufficient PGV token voting power at the start of the voting period.

Instead of using the current token balance, the contract retrieves the user's historical voting power using getPastVotes, capturing a snapshot of their balance at the proposal's start time. This ensures consistency and prevents manipulation through token transfers after the proposal's creation.

```
function castVote(bytes32 proposalId, bool support) external {
    Proposal storage proposal = proposals[proposalId];
    if (proposal.startTime == 0) revert ProposalNotFound(proposalId);
    require(block.timestamp < proposal.endTime, "Voting period ended");
    require(!hasVoted[proposalId][msg.sender], "Already voted");

    uint256 votingPower = governanceToken.getPastVotes(msg.sender,proposal.startTime);
    require(votingPower > 0, "No voting power");

    if (support) {
        proposal.yesVotes += votingPower;
    } else {
        proposal.noVotes += votingPower;
    }

    hasVoted[proposalId][msg.sender] = true;
    emit Voted(proposalId, msg.sender, support, votingPower);
}
```

Figure 21: Code - castVote function

Upon casting a vote, the user's voting power is added to either the yesVotes or noVotes count of the proposal, based on their choice. The system restricts users from voting more than once per proposal, preserving the integrity of the voting process.

7.3.5. Secondary chain vote collection

After the voting period concludes on secondary chains, the relayer automatically finalizes the voting process by calling the finalizeVotes function on each SecondaryGovernance contract. Once finalized, the secondary chain emits a VotesTallied event containing the final tallies (yesVotes and noVotes).

The relayer, which continuously listens for VotesTallied events via WebSocket, detects this event in real-time. Upon detection, it initiates the vote collection process by calling the collectSecondaryChainVotes function on the MainGovernance contract.

```
function collectSecondaryChainVotes(
    bytes32 proposalId, string memory chainId, uint256 yesVotes, uint256 noVotes
) external onlyRelayer {
    require(registeredChains[chainId], "Chain not registered");
    require(
        block.timestamp >= proposals[proposalId].endTime, "Voting period not ended"
    );
    require(
        !secondaryChainVotes[proposalId][chainId].collected, "Votes already collected"
    );

    secondaryChainVotes[proposalId][chainId] = VoteSummary({
        yesVotes: yesVotes,
        noVotes: noVotes,
        collected: true
    });

    emit SecondaryChainVotesCollected(
        proposalId,
        chainId,
        yesVotes,
        noVotes
    );
}
```

Figure 22: Code - collectSecondaryChainVotes function

On the relayer side, it initially queries the main chain to determine if votes from the secondary chain have already been collected by reading the secondaryChainVotes mapping. If the votes are uncollected, it submits the vote counts via a transaction to the main governance contract.

```
for (const [chainId, { contract }] of Object.entries(secondaryConnections)) {
    contract.on("VotesTallied", async (proposalId: string) => {
        console.log(`📬 Votes tallied for proposal ${proposalId} on ${chainId}`);
        const secondaryGov = new SecondaryGovernance(contract);

        try {
            const secondaryProposal = await secondaryGov.getProposalDetails(proposalId);

            await collectVotesFromSecondary(
                mainGov,
                proposalId,
                chainId,
                secondaryProposal.yesVotes,
                secondaryProposal.noVotes
            );
```

Figure 23: Code - Relayer - Listening to VotesTallied Event

This mechanism ensures that each secondary chain's voting results are securely synchronized with the main chain and prevents duplicate collection for the same proposal.

7.3.6. Finalization

Once the voting period ends and votes from all secondary chains are collected, the finalizeProposalVotes function is called on the main chain. This function first verifies that the voting period has ended. It then aggregates the votes from the main chain and all registered secondary chains (via the secondaryChainVotes mapping). The final tallies are saved into the proposal's finalYesVotes and finalNoVotes fields, and the proposal is marked as finalized by setting voteTallyFinalized to true.

Based on the aggregated results, the proposal status is updated to either Accepted or Rejected. If the total number of votes does not meet the quorum requirement, the proposal is automatically rejected. If the proposal is accepted, it becomes eligible for execution. The executeProposal function can then be called separately to trigger any attached on-chain actions.

```
function finalizeProposalVotes(bytes32 proposalId) external nonReentrant {
    Proposal storage proposal = proposals[proposalId];
    if (proposal.startTime == 0) revert ProposalNotFound(proposalId);
    if (block.timestamp < proposal.endTime) revert("Voting period not ended");
    if (proposal.voteTallyFinalized) revert ProposalAlreadyFinalized(proposalId);

    uint256 totalYesVotes = proposal.yesVotes;
    uint256 totalNoVotes = proposal.noVotes;

    for (uint i = 0; i < chainList.length; i++) {
        string memory chainId = chainList[i];
        if (!secondaryChainVotes[proposalId][chainId].collected) {
            revert VotesNotCollected(chainId);
        }

        VoteSummary storage summary = secondaryChainVotes[proposalId][chainId];
        totalYesVotes += summary.yesVotes;
        totalNoVotes += summary.noVotes;
    }

    uint256 totalVotes = totalYesVotes + totalNoVotes;
    proposal.finalYesVotes = totalYesVotes;
    proposal.finalNoVotes = totalNoVotes;
    proposal.voteTallyFinalized = true;

    if (totalVotes < quorumVotes) {
        proposal.status = ProposalStatus.Rejected;
    } else {
        if (totalYesVotes > totalNoVotes) {
            proposal.status = ProposalStatus.Accepted;
        } else {
            proposal.status = ProposalStatus.Rejected;
        }
    }

    emit VoteTallyFinalized(proposalId, totalYesVotes, totalNoVotes);
}
```

Figure 24: Code - finalizeProposalVotes function

This finalization step ensures that the final result accurately reflects votes collected across all chains, achieving a consistent and synchronized governance outcome.

7.3.7. Proposal execution

After vote finalization, the proposal outcome is determined based on the aggregated votes. If the final yesVotes exceed the noVotes, and quorum requirements are met, the proposal status is set to Accepted. The executeProposal function must then be manually called to execute the proposal.

During execution, the contract iterates through each of the proposal's targets, values, and calldatas, performing low-level calls to execute the specified actions. If all target calls succeed, the proposal is marked as Executed.

If the proposal was previously rejected (due to lack of quorum or more no votes), execution is not allowed, and the proposal remains in the Rejected state.

```
function executeProposal(bytes32 proposalId) external nonReentrant {
    Proposal storage proposal = proposals[proposalId];
    if (!proposal.voteTallyFinalized) revert("Votes not finalized");
    if (proposal.status == ProposalStatus.Executed)
        revert ProposalAlreadyExecuted(proposalId);
    if (proposal.status != ProposalStatus.Accepted)
        revert("Proposal not accepted");

    for (uint i = 0; i < proposal.targets.length; i++) {
        (bool success, ) = proposal.targets[i].call{
            value: proposal.values[i]
        }(proposal.calldatas[i]);
        if (!success) revert ExecutionFailed(proposal.targets[i], i);
    }

    proposal.status = ProposalStatus.Executed;

    emit ProposalExecuted(proposalId, proposal.status);
}
```

Figure 25: Code - executeProposal function

This mechanism ensures that only approved governance actions are enforced on-chain, maintaining decentralized and verifiable decision-making.

7.4. User interface

The user interface was developed using Next.js to ensure an intuitive and user-friendly interaction with the governance system. Key components include **Proposal Dashboard**, **Voting Interface** and **Proposal Creation Form** (Appendix G: High Fidelity UI Design). These ensure seamless user interaction with core functionalities while maintaining a clean and responsive design.

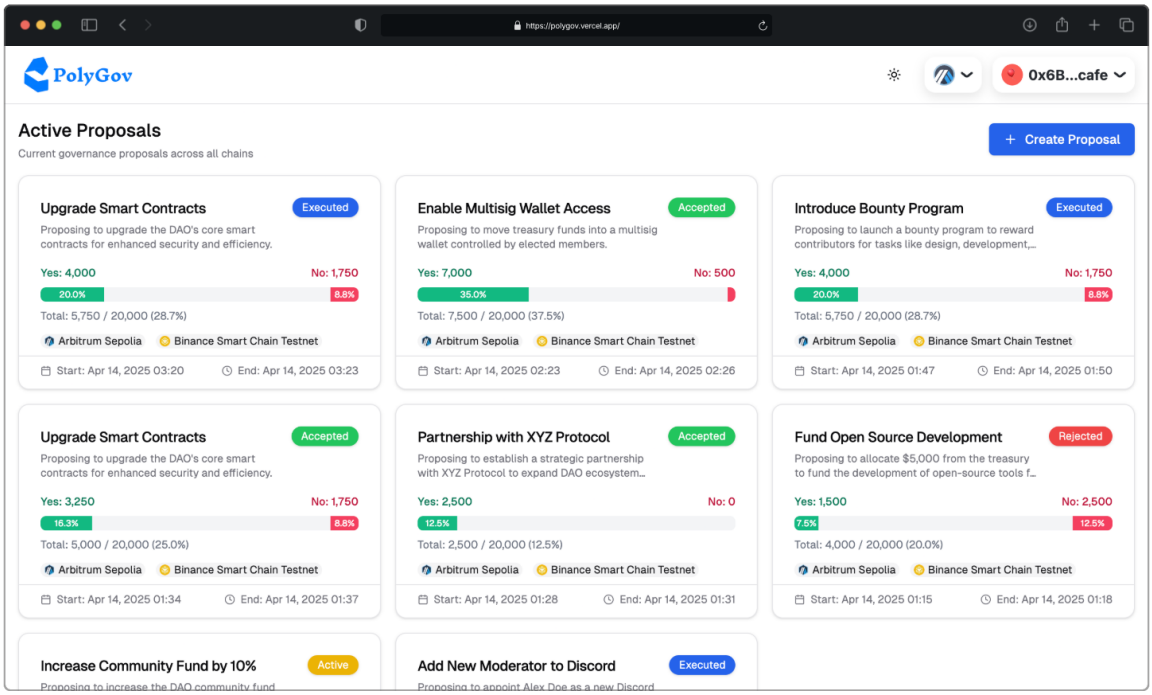


Figure 26: UI - Proposal Dashboard

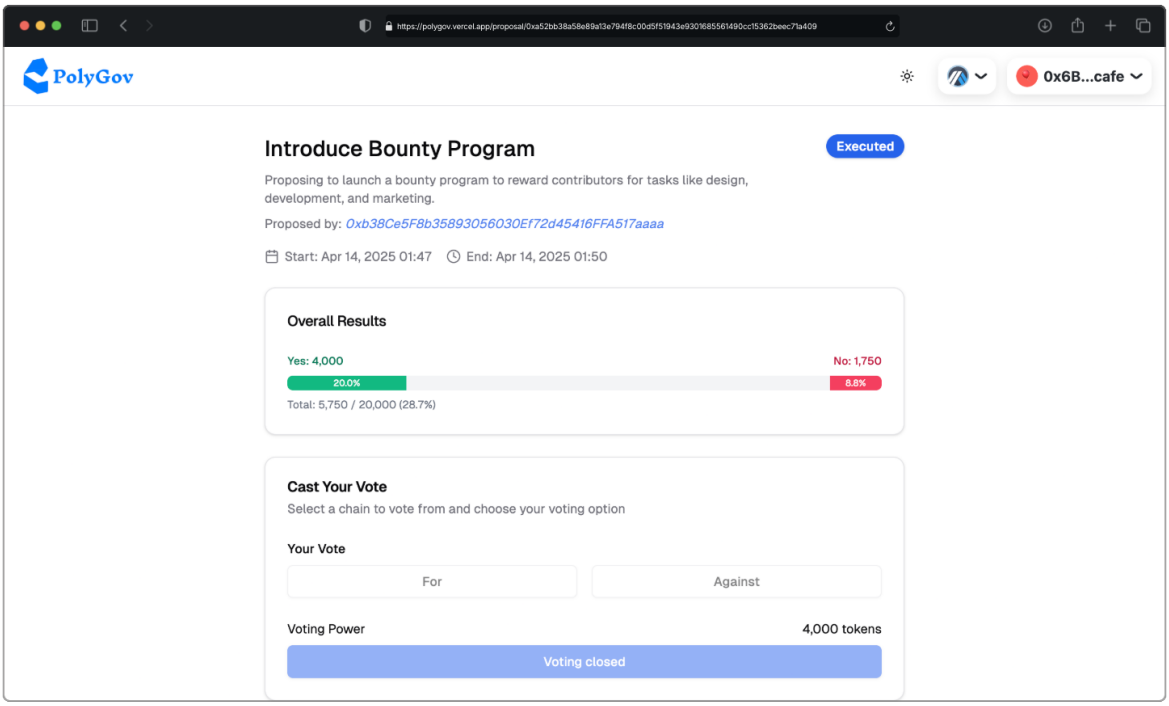


Figure 27: UI - Voting Interface

7.5. Chapter summary

This chapter has explained how the multi-chain governance voting system was implemented through the integration of smart contracts, relayer logic, and user interface components. The core functionalities, including proposal creation, mirroring, voting, vote aggregation, and execution, were developed to support decentralized and synchronized governance across multiple chains. The next chapter will evaluate the system's performance and explore potential areas for improvement.

CHAPTER 08: TESTING

8.1. Chapter overview

The primary purpose of testing is to verify that the multi-chain governance voting system operates correctly and meets the project requirements. This chapter outlines the overall testing approach, the types of testing performed, and a summary of the results. Testing focused on key areas such as proposal creation, voting, cross-chain synchronization, and system security to ensure the system meets the requirements and performs correctly.

8.2. Objectives and goals of testing

The primary objective of testing in this research is to ensure that the developed multi-chain governance voting system performs according to the defined requirements. The specific goals of the testing process are:

- To verify that all the core modules operate as intended across all tested blockchain networks.
- To confirm that the system meets all critical ("Must Have") and important ("Should Have") requirements specified using the MoSCoW prioritization method.
- To confirm that the relayer service accurately synchronizes across chains without introducing data loss, duplication, or inconsistencies.
- To assess whether the deployed smart contracts correctly enforce governance rules, including quorum thresholds, proposal state transitions, and accurate vote tallying.
- To evaluate the system against key non-functional requirements such as scalability, transparency, security and reliability.
- To identify any functional limitations, usability challenges, or security vulnerabilities in both the on-chain smart contracts and the off-chain relayer service that may require future improvements or optimizations.

8.3. Testing criteria

The system testing was designed using two primary dimensions to evaluate its performance, functionality, and structural integrity. These testing criteria aim to bridge the gap between system expectations and actual implementation:

1. **Functional Quality:** This focuses on verifying that the core features of the system including user-facing operations (e.g., proposal creation, voting) and backend processes (e.g., proposal mirroring, vote finalization, execution) behave according to the functional requirements defined in the SRS.
2. **Integration and Structural Quality:** This involves assessing the system's ability to correctly synchronize proposals and votes across the main and secondary chains through the relayer. It ensures that smart contracts interact correctly across modules (e.g., governance, relayer, execution) and maintain secure and reliable behaviour.

8.4. Unit Testing

Unit testing was carried out to validate the functionality, reliability, and correctness of the smart contracts developed for the decentralized multi-chain governance system. All core smart contracts were tested, including **PGVToken.sol**, **MainGovernance.sol**, **SecondaryGovernance.sol**, and **DemoContract.sol**. Unit testing was conducted using Hardhat as the testing framework.

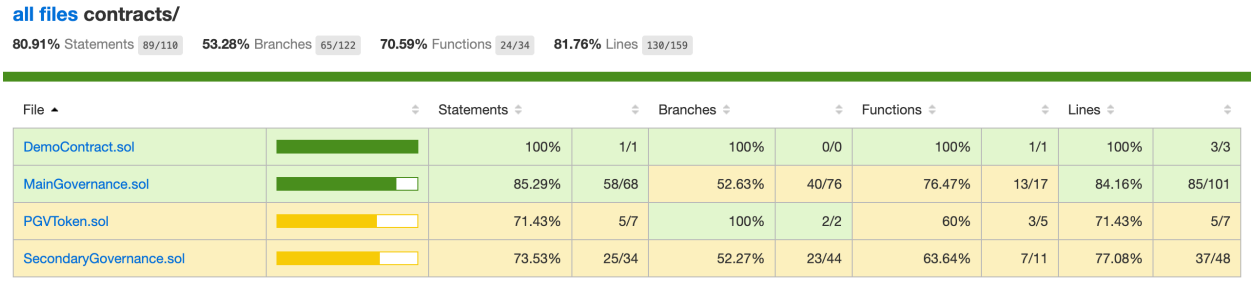


Figure 28: Hardhat code coverage analysis

Code coverage analysis showed that more than 70% of the statements, functions, and lines were successfully covered during testing. The uncovered areas mainly correspond to low-priority or auxiliary functions, which do not significantly impact the core system functionality.

The results of the unit testing process are summarized as follows:

- **PGVToken Tests:** The PGVToken contract passed all major tests, including token minting, auto-delegation of votes, and correct voting power assignment after delegation. Tests confirmed that vote delegation functions properly when tokens are transferred and that no auto-delegation occurs during contract transfers.
- **MainGovernance Tests:** The MainGovernance contract was extensively tested for proposal creation, voting, quorum updates, and secondary chain registration. The contract correctly handled proposal validation, voting timelines, and permission restrictions. Voting after the proposal end time and double voting scenarios were successfully reverted as expected.
- **SecondaryGovernance Tests:** The SecondaryGovernance contract was tested for mirroring proposals, handling vote submissions, finalizing voting, and relayer validation. The contract accurately reverted invalid mirroring attempts and disallowed unauthorized relayer actions.
- **End-to-End Flow Tests:** Full end-to-end flow testing was conducted, covering proposal creation, mirroring, voting, finalization, and execution. The system correctly rejected proposals that did not meet quorum and prevented actions like executing a rejected proposal or voting without secondary chain results.

Figure 29: Unit test results

8.5. Functional testing

To ensure the system meets both its functional requirements and is resilient under different conditions, functional testing was designed to cover standard use cases, error handling, and edge cases. Test cases are based directly on the defined functional requirements but are extended to include additional scenarios.

Test Case	Description	Input	Expected Outcome	Status
TC-F01	Create proposal on main chain	Valid proposal details	Proposal created successfully and event emitted	✓ TxHash
TC-F02	Vote on main chain proposal	Valid vote by user	Vote recorded correctly with voting power	✓ TxHash
TC-F03	Vote on secondary chain proposal	Valid vote by user	Vote recorded correctly on secondary chain	✓ TxHash
TC-F04	Mirror proposal to secondary chain	Proposal created event emitted	Proposal mirrored successfully by relayer	✓ TxHash
TC-F05	Collect and submit secondary chain votes	Finalized votes submitted	Main chain records secondary votes correctly	✓ TxHash
TC-F06	Finalize proposal after votes collected	Proposal voting ended	Proposal finalized (Accepted/Rejected)	✓ TxHash
TC-F07	Execute accepted proposal	Proposal passed	Execution calls successful and event emitted	✓ TxHash
TC-F08	Unauthorized action prevention	Non-relayer tries to mirror votes	Transaction reverted	✓
TC-F09	View proposal details and voting results	Query proposal info	Proposal details and status returned correctly	✓

Table 13: Functional testing

8.6. Module and integration testing

Module and integration testing ensure that the different components of the system, such as smart contracts, relayer scripts, and front-end interactions, operate correctly both individually and when combined as a complete system.

Module	Sub-Components	Integration Points	Test Description	Status
Governance Token	Mint, Transfer, Delegate	Token → Governance Contracts	Ensure correct voting power	✓
Main Governance	Proposal Creation, Voting, Finalization, Execution	User Wallet → Main Governance	Users create / vote / finalize / execute proposals	✓
Secondary Governance	Mirror Proposal, Voting, Finalization	Main Governance ↔ Secondary Governance (Relayer)	Relayer mirrors proposals and collects votes	✓
Relayer Logic	Mirror, Collect, Finalize	Relayer → Secondary Governance / Main Governance	Relayer mirrors and collects votes reliably	✓
Demo Contract Execution	Proposal Execution	Main Governance → Demo Contract	Executing proposals triggers external contract actions	✓

Table 14: Module and integration testing

8.7. Non-functional testing

Non-functional testing was carried out to evaluate the performance, usability, scalability, and reliability of the decentralized governance system. These tests focused on how the system behaves under different conditions rather than just verifying individual functionalities.

Test Case	Description	Expected Outcome	Status
TC-NF01	Data consistency across chains	Proposal data and vote results are accurately mirrored and synchronized across all connected blockchains.	✓

TC-NF02	Unauthorized access prevention	Only authorized users (proposal creators, voters) and the registered relayer can perform governance actions; unauthorized attempts are rejected.	✓
TC-NF03	Resilience to network disruption	The system remains stable during temporary network disconnections. On reconnection, pending operations resume correctly without data loss.	✓
TC-NF04	User Interface usability	The web-based dApp provides a clear and easy-to-use interface for proposal creation, voting, and tracking proposal statuses.	✓
TC-NF05	Response time under load	Voting and proposal creation operations respond within 5 seconds even under multiple simultaneous transactions.	✓
TC-NF06	Scalability with new chains	New secondary chains can be added dynamically without needing major system configurations. Governance actions work across added chains.	✓

Table 15: Non-functional testing

8.8. Performance Testing

Appendix H: Performance Testing includes an overview of the performance tests carried out for the PolyGov platform. Vercel Insights monitored real-time metrics in the production environment, tracking load times, server responses, and user interactions. Google Lighthouse audits assessed performance, accessibility, best practices, SEO, and PWA compliance.

8.9. Limitations of the testing process

- **Testnet Constraints:** Restrictions such as faucet limitations and token shortages on testnets reduced the ability to perform a higher volume of complete governance tests.
- **Testnet Environment Limitation:** Testing was conducted on local blockchain networks and public testnets, which may not fully replicate the operational conditions of a live mainnet environment. Factors such as network congestion, miner behaviours, or real-world relayer failures were not fully simulated.

- **Security Auditing Scope:** Although security testing focused on major vulnerabilities such as unauthorized access, vote manipulation, and replay attacks, a full formal audit (e.g., by a third-party security firm) was not conducted due to resource constraints.
- **Scalability Testing:** The system's scalability was evaluated under controlled conditions. However, large-scale performance testing involving dozens of secondary chains and hundreds of simultaneous proposals was not practically achievable within the available infrastructure.
- **Eventual Consistency under Network Failures:** While tests simulated temporary network interruptions, long-term blockchain forks or relayer downtime scenarios were not exhaustively tested over prolonged periods.

8.10. Chapter summary

This chapter described the testing activities carried out to validate both the individual components and the overall behaviour of the multi-chain governance voting system. Functional, module, and non-functional tests confirmed that the smart contracts and relayer logic worked as expected according to the system requirements. Limitations of the testing process, such as manual testing, limited stress testing, and testnet environment limitations were acknowledged. Overall, the testing provided strong evidence of the system’s correctness. However, full reliability at larger scale would require future real-world deployment and broader community-based testing.

CHAPTER 09: EVALUATION

9.1. Chapter overview

This chapter presents the evaluation of the implemented multi-chain governance voting system. It outlines the methodology used to assess the effectiveness of the system, defines the criteria for evaluation, includes a self-assessment of the research process, and summarizes the feedback gathered from selected evaluators. The chapter also highlights the limitations encountered during the evaluation phase.

9.2. Evaluation methodology and approach

The evaluation of the proposed multi-chain governance voting system was conducted using a combination of practical testing and qualitative analysis. The system was deployed across multiple EVM-compatible testnets to simulate real-world scenarios within a controlled environment. These testnets were selected to reflect operational challenges such as fluctuating gas fees, block confirmation delays, and asynchronous communication between blockchains.

In addition to technical testing, qualitative feedback was collected from experts in blockchain development and decentralized governance. This feedback was used to assess the system's usability, scalability, security, and alignment with decentralized governance principles.

By combining hands-on technical testing with stakeholder input, the evaluation provided a comprehensive understanding of the system's strengths, limitations, and potential areas for future improvement.

9.3. Evaluation criteria

Criteria	Objective
Problem Significance and Relevance	To assess whether the identified problem in cross-chain governance is important and justifies a technical solution.
Technical Scope and System Complexity	To evaluate whether the project meets the expected complexity, depth, and breadth appropriate for undergraduate-level research.

Design and Innovation of the Solution	To evaluate the decisions made during the design and development phases, and to examine the uniqueness of the proposed system in terms of reusability and automation.
System Functionality and Performance	To determine whether the system functions correctly under realistic test conditions and meets the intended goals of automation, and cross-chain consistency.
Limitations and Future Improvements	To identify any limitations in the current implementation and provide suggestions for future enhancements that can improve the systems performance.

Table 16: Evaluation criteria

9.4. Self-evaluation

Criteria	Evaluation
Problem Significance and Relevance	The research addressed a meaningful gap in decentralized governance by proposing a scalable, automated multi-chain voting solution. Literature findings and stakeholder feedback confirmed the problem's relevance in both academic and industry settings.
Technical Scope and System Complexity	The project involved multiple components, including smart contracts, relayer logic, WebSocket event handling, and cross-chain deployment. The technical depth and architectural complexity were appropriate for a final-year undergraduate project.
Design and Innovation of the Solution	The reusable governance architecture and cross-chain proposal mirroring mechanism introduced a novel approach to addressing existing inefficiencies observed in frameworks like MULTAV. Although the implementation was limited to EVM-compatible testnets, the design remains extensible to other blockchain ecosystems.
System Functionality and Performance	The system was successfully tested on appropriate testnets, demonstrating correct proposal synchronization and vote aggregation. However, certain areas such as gas cost optimization and

	Solana integration remain incomplete, highlighting opportunities for future refinement.
Limitations and Future Improvements	Several limitations were identified, including reliance on a centralized relayer, limited validator diversity, and incomplete multi-chain coverage. These were documented clearly and used to guide future improvement plans, including implementing decentralized relayer mechanisms and expanding to non-EVM chains.

Table 17: Self-evaluation

9.5. Selection of the evaluators

The evaluators were selected based on their expertise in blockchain development, decentralized governance, and smart contract auditing. The panel included core team members from blockchain communities, experts from blockchain organizations, and technical professionals with experience in development, security, and quality assurance. Each evaluator provided valuable feedback on the system's functionality, scalability, and security, with a focus on ensuring decentralization.

The selected evaluators are categorized into two groups: Domain Experts and Technical Experts, as shown below:

ID	Name	Position	Affiliation
Domain Experts			
EV1	N. H. K. Asanka	Core Team Member and Validator	PostHuman Validator (posthuman.digital)
EV2	Aman Ashraf	Binance Angel	Binance
EV3	Okeesha Vinula	Confidential	Sputnik Network
Technical Experts			
EV4	Nisal Chadrasekara	CEO	Ceyloncash
EV5	Dilshan Madushanka	Blockchain developer CTO	Bake.io CeylonCash
EV6	Geeth Gayashan	Quality Assurance Engineer	Spera Labs

Table 18: Selection of the evaluators

9.6. Evaluation result

This provides a summary of the evaluations conducted by domain and technical experts. The complete evaluation details can be found in the **Appendix I: Evaluations of Domain & Technical experts**.

Criteria	Summary of Feedback
Problem Significance and Relevance	Experts agreed that the PolyGov project addresses an important and timely problem in the field of multichain DAO governance. They consider the project highly relevant for the future of decentralized systems and believe it provides an essential solution for cross-chain protocol governance.
Technical Scope and System Complexity	The technical scope is suitable for the current stage of development. The project successfully manages multichain voting and proposal processes. However, experts noted that additional work is needed to automate cross-chain execution and to fully decentralize the system for future use.
Design and Innovation of the Solution	The system design was seen as clean, modular, and practical. Experts appreciated the flexible architecture and the ability to adjust governance logic for different blockchains. However, they pointed out that reliance on a centralized relayer could become a weakness if not improved in later versions.
System Functionality and Performance	The system performs well at the demonstration stage. Core features are working as expected, particularly across EVM-compatible chains and Layer 2 solutions. Experts noted that the system will need further improvements in scalability, and that off-chain components should be carefully monitored.
Limitations and Future Improvements	Experts suggested several improvements. These include reducing the trust placed in the relayer by using decentralized or multisignature methods, supporting non-EVM and UTXO-based blockchains, adding verifiable audit logs for off-chain operations, exploring technologies like LayerZero, and conducting thorough security audits before moving to production.

Table 19: Evaluation results

9.7. Limitations of evaluation

- **Limited Number of Evaluators:** Feedback was collected from only a small group of evaluators, primarily blockchain developers and domain experts. A larger and more diverse group, including DAO members or governance researchers, could have provided broader perspectives.
- **Testnet-Based Evaluation:** The system was evaluated exclusively on EVM-compatible testnets. Testing on real mainnets or under live DAO governance environments was not conducted, limiting the understanding of how the system would perform under full decentralization and real-world conditions.
- **Short Evaluation Period:** Due to time constraints, evaluators got to know about the system over a short period. Longer-term usage could have revealed additional usability or performance issues not captured during initial testing.
- **No Formal Security Audit:** The system underwent individual security checks but did not receive an independent third-party smart contract audit, limiting the formal evaluation of system security.

9.8. Evaluation on functional requirements

IM: Implemented | **NI**: Not Implemented

FR ID	Requirement	Priority	Status
FR01	Users must be able to create governance proposals on the main blockchain, specifying title, description, voting period, and executable actions.	M	IM
FR02	Users must be able to vote on active proposals on both main and secondary blockchains, weighted by their voting power.	M	IM
FR03	The system must finalize proposals after voting ends, aggregating votes from the main and secondary chains securely.	M	IM
FR04	The relayer must mirror proposals from the main chain to secondary chains automatically via events.	M	IM

FR05	The relayer must collect finalized votes from secondary chains and submit them to the main chain for aggregation.	M	IM
FR06	The system should allow users to view proposal details, status (pending, accepted, rejected, executed), and vote tallies.	S	IM
FR07	The system could execute accepted proposals by performing predefined external contract actions.	C	IM
FR08	The system should prevent unauthorized actions (e.g., non-relayer trying to mirror or finalize votes).	S	IM
FR09	The system could allow users to delegate voting power to others (proxy voting).	C	NI
FR10	The system could integrate with external systems (oracles, off-chain data feeds) to enhance governance decisions.	W	NI

Table 20: Evaluation on functional requirements

9.9. Evaluation on non-functional requirements

IM: Implemented | **NI**: Not Implemented

ID	Requirement	Priority	Status
NFR01	The system must ensure data consistency across the main and secondary chains, keeping proposals and votes synchronized.	M	IM
NFR02	The system must protect against unauthorized access, ensuring that only eligible users can create proposals or cast votes.	M	IM
NFR03	The system must be resilient to network disruptions (e.g., chain downtime or fork events) without compromising the voting process.	M	IM
NFR04	The system should provide a user-friendly interface for interacting with proposals, voting, and monitoring outcomes.	S	IM
NFR05	The system should maintain an average response time of less than 5 sec for user actions such as creating a proposal or voting.	S	Reasonable speed achieved
NFR06	The system should be scalable to support additional secondary chains without major changes to the architecture.	S	IM

NFR07	The system could provide real-time alerts and notifications for major governance events (e.g., proposal created, voting ended).	C	NI
NFR08	The system support analytics and reporting features (e.g., voter turnout, proposal success rate).	W	NI

Table 21: Evaluation on non-functional requirements

9.10. Chapter summary

This chapter outlined the evaluation process used to assess the effectiveness, quality, and completeness of the developed multi-chain governance voting system. A combination of practical testing on EVM-compatible testnets and qualitative feedback from domain experts and technical evaluators was used to validate the system's functionality. While the system met most functional objectives, certain limitations such as centralized relayer dependency and incomplete decentralization highlight important areas for future improvement. Overall, the evaluation confirmed that the project meets the academic standards expected of a final-year research project and offers meaningful contributions toward improving decentralized governance in blockchain ecosystems.

CHAPTER 010: CONCLUSION

10.1. Chapter overview

This chapter concludes the research by reflecting on the aims and objectives achieved, the knowledge applied from academic learning and industry experience, and the new technical skills acquired. It also presents the challenges encountered during the development of the decentralized multi-chain governance voting system, deviations from the original plan, research limitations, and future directions for enhancement. Finally, it assesses the contribution this research has made to the broader field of blockchain governance and decentralized systems.

10.2. Achievements of research aims & objectives

10.2.1. Achievement of aim

The aim of this research is to design and develop a scalable and automated multi-chain governance system that focuses specifically on proposal management and voting synchronization across multiple blockchain networks.

The research successfully achieved this aim by proposing and implementing a reusable governance architecture supported by modular smart contracts and an off-chain relayer mechanism. The final system enables cross-chain proposal mirroring, automated vote collection, aggregation, and consistent status updates across different blockchain networks.

Evidence of the system's ability to automate proposal management and ensure synchronized voting outcomes is demonstrated through the detailed implementation in CHAPTER 07: and the testing results presented in CHAPTER 08:.

10.2.2. Achievement of objectives

Research Objective	Status
Problem Identification: An in-depth review of existing blockchain governance practices was conducted to identify inefficiencies in multi-chain governance systems.	Completed

Literature Review: A comprehensive literature review was performed, covering traditional and blockchain-based governance models, cross-chain synchronization methods, and various voting mechanisms	Completed
Requirement Analysis: Stakeholder requirements were collected using surveys and analyzed to define the functional and non-functional needs of a multi-chain governance system.	Completed
System Design: A modular and reusable smart contract architecture was designed to support dynamic proposal creation and voting across multiple chains.	Completed
Implementation: The system was implemented using Solidity for smart contracts and TypeScript for the relayer logic based on designed architecture.	Completed
Testing and Evaluation: The implemented system was evaluated based on security, decentralization, transparency, and gas efficiency.	Completed
Documentation: All phases of the project from problem domain, literature, design, and implementation to evaluation were documented in this thesis.	Completed
The system and findings are prepared for formal presentation and in the VIVA, where critical feedback and questions will be addressed.	Pending

Table 22: Achievement of objectives

10.3. Utilization of knowledge from the course

Although the curriculum did not directly cover blockchain development or decentralized systems, several core concepts and skills learned during the course were effectively applied throughout the project:

1. **Object-Oriented Programming (OOP):** OOP principles were applied to structure the smart contracts and relayer logic in a modular and maintainable way.
2. **Web Development:** Skills in JavaScript, CSS, and front-end technologies were used to build the user interface for proposal creation, voting, and result tracking.
3. **Software Development I and II:** Foundational programming skills gained from Java and Python modules helped in understanding complex logic structures and implementing both on-chain and off-chain components.

4. **Client-Server Architecture:** Knowledge of client-server models and WebSocket communication was utilized to design the event-driven relayer for cross-chain proposal synchronization.
5. **Software Development Group Project:** Experience gained from this module was essential for applying best practices in software engineering, including problem identification, requirement analysis, system design, development, testing, and documentation all over the thesis.

10.4. Use of existing skills

Several existing skills, developed through industry experience and prior learning, were applied during this project:

- **Industry Experience:** Previous exposure to full-stack development and API integration provided the confidence to manage the interaction between smart contracts (on-chain) and the relayer service (off-chain).
- **Blockchain Basics:** Foundational knowledge of blockchain concepts, such as decentralized networks, consensus mechanisms, and token standards, helped in understanding the design and deployment of governance smart contracts.
- **Front-End Development:** Experience with modern front-end frameworks and routing systems enabled the rapid development of the governance dApp, improving both development speed and user experience.

10.5. Use of new skills

This project required the acquisition and application of several new technical skills beyond the existing knowledge base:

- **Smart Contract Development:** Solidity was learned from scratch and gained hands-on experience in writing, testing, and deploying smart contracts, focusing on governance-specific functionalities such as voting, proposal management etc.
- **Cross-Chain Communication:** Developed an understanding of event-driven architecture for blockchain interoperability by building a relayer that mirrors proposals and synchronizes votes across multiple blockchain networks.

- **Blockchain Deployment and Testing:** Acquired practical knowledge of deploying decentralized applications across multiple EVM-compatible testnets and managing network-specific configurations.
- **Security Considerations in Blockchain Systems:** Gained awareness of common vulnerabilities in smart contracts and the importance of implementing security best practices, even though formal auditing was beyond the project scope.

10.6. Achievement of learning outcomes

Learning Outcome	What has been learned
LO1: Apply appropriate methods, techniques, and tools for a large problem	Used blockchain technology with libraries and tools like Solidity, Hardhat, and ethers.js to design and implement a reusable, scalable multi-chain governance system.
LO2: Develop a project plan and manage time effectively	Followed a structured project timeline with defined milestones covering research, design, development and testing phases.
LO3: Collect and analyze project requirements	Gathered and analyzed system requirements through stakeholder input and documented them in the SRS.
LO4: Research and critically evaluate relevant information	Conducted a comprehensive literature review to understand the strengths and limitations of existing governance voting models and cross-chain communication technologies.
LO5: Work autonomously to gain new skills	Independently learned Solidity, smart contract development, and cross-chain relayer scripting through self-study, tutorials, and practical experimentation.
LO6: Address EDI, legal, ethical, and professional issues	Ensured compliance with licensing (MIT License), handled survey participation ethically, and properly cited all external research sources.
LO7: Produce extended practical work	Delivered smart contracts, relayer scripts, and a working front-end dApp as part of the system.
LO8: Produce a coherent and evaluative report	Documented implementation, challenges, evaluation, and reflections in the final report.

LO9: Defend the work at a viva voce examination	Will be presented and defended through a formal viva, explaining the research and implementation.

Table 23: Achievement of learning outcomes

10.7. Problems and challenges faced

Challenge	Description	Solution
High Gas Costs	Layer 1 testnets had expensive gas fees, limiting testing.	Moved to Arbitrum Sepolia for cost-efficient testing.
Faucet Limitations	Faucets had minimum mainnet token balance requirements and limited access.	Purchased small amounts of mainnet tokens and claimed the faucets frequently.
Knowledge Gap	Initial lack of experience in smart contracts and relayers.	Followed advanced tutorials and studied documentation.
WebSocket Rate Limiting	Frequent reconnections during relayer triggered rate limits on node provider (Alchemy)	Limited WebSocket usage only to the testing phase and disabled it when not actively needed.
Unstable WebSocket Connections	Event listeners frequently disconnected, missing updates.	Built a health-check and auto-reconnect mechanism.
Gas Cost Inaccuracy	Estimators underreported actual costs, causing deployment issues.	Manually optimized contracts and tested on mainnet, but some issues remained (Contract).
Multi-Chain Testing Limitations	Tools lacked proper cross-chain simulation capabilities.	Conduct every test on testnet environments which required to deploy the contract every time.

Table 24: Problems and challenges faced

10.8. Deviations

During the project, there were no major deviations from the initial planning and system design. However, some changes occurred in the technology selection due to technical and practical challenges:

Solana Integration Abandoned

The project planned to extend support to both SVM (Solana) and EVM-compatible blockchains. Basic voting and proposal functionality was prototyped using Solana's Rust-based development stack. However, several challenges were encountered, such as limited tooling support compared to EVM ecosystems, high development complexity due to Rust programming requirements, lack of mature governance features and supporting community on Solana, and the author's limited experience with Rust and Solana concepts, which further slowed progress. Due to these constraints, and in order to maintain focus and meet project deadlines, Solana integration was discontinued. Partial Solana code attempts are documented in **Appendix K: Solana Integration Draft** for future reference and potential continuation.

Migration from Sepolia to L2 Testnet

Ethereum Sepolia was first used for development, but faucet, RPC issues as shown in **Appendix L: Sepolia Issue** and high gas fees created major delays. To enable faster and more cost-effective testing, the project shifted to Layer 2 testnets such as Arbitrum Sepolia. This improved deployment speed, allowed more frequent tests, and reduced development overhead.

10.9. Limitations of the research

While the research successfully demonstrated a scalable and automated multi-chain governance system for proposal management and voting, several limitations were encountered:

- **Relayer Centralization:** The relayer component used in the system was developed as a single centralized service for simplicity. A production-ready governance system would require a decentralized or more fault-tolerant relayer network to ensure trustlessness.
- **Limited Cross-Chain Diversity:** Only EVM-compatible blockchains were tested. Integration with non-EVM blockchains like Solana, Cosmos, or Polkadot remains unimplemented.
- **Security Evaluation:** While basic internal security checks were performed, no formal third-party audit was conducted on the smart contracts or relayer code.

10.10. Future enhancements

Several future enhancements are proposed to improve the scalability, security, and decentralization of the multi-chain governance voting system:

- **Decentralized Relayer Network:** Replace the centralized relayer with a decentralized network of relayers. Techniques such as interoperability technologies such as LayerZero, multi-signature validation, Merkle-proof or on-chain relayer incentivization could be implemented to eliminate single points of failure and enhance trustlessness in cross-chain synchronization.
- **Advanced Governance Controls:** Introduce features such as proposal rejection, deletion, emergency pause mechanisms and a Timelock execution. These controls would be enforced through quorum-approved governance logic, providing greater flexibility and improving the system's ability to respond to malicious or accidental proposals.
- **Integration with Non-EVM Blockchains:** Expand interoperability beyond EVM-compatible networks by integrating support for non-EVM blockchains such as Solana, Cosmos, and Polkadot. This would broaden the system's applicability across diverse blockchain ecosystems and improve its relevance in the evolving multi-chain environment.

10.11. Achievement of the contribution to body of knowledge

This project successfully achieved the intended contributions to both the research domain and the problem domain.

10.11.1. Contribution to the problem domain

The project developed a working multi-chain governance system that allows proposals and voting processes to be mirrored, synchronized, and finalized across different blockchains.

By automating these tasks and reducing manual work, the system improved scalability and efficiency compared to existing solutions like MULTAV.

The web-based decentralized application (dApp) also made governance participation easier for users by providing real-time proposal tracking, graphical voting results, and wallet integration, helping to create a more inclusive and accessible environment for decentralized decision-making.

10.11.2. Contribution to the research domain

This research successfully demonstrated how a trusted relayer can be securely used to manage cross-chain proposal and voting synchronization in decentralized governance.

It introduced a modular and reusable smart contract design that reduced the deployment overhead typically seen in cross-chain systems.

The project also expanded academic knowledge in the area of blockchain governance by presenting a scalable and automated framework for multi-chain environments, supported by testing, evaluation, and analysis throughout the implementation.

10.12. Concluding remarks

This project has demonstrated that a secure, transparent, and scalable multi-chain governance system can be achieved through the careful integration of smart contracts and a trusted relayer mechanism. Despite certain limitations, the developed prototype successfully synchronized voting processes across multiple blockchains, providing a decentralized foundation for future on-chain governance applications.

The research journey significantly enriched both technical and academic skills, contributing to a deeper understanding of decentralized systems and cross-chain interoperability. The findings of this project offer a solid groundwork for future advancements in decentralized governance, particularly as blockchain ecosystems continue to evolve toward more interconnected and interoperable architectures.

REFERENCES

Ao, Z. et al. (2023). Is decentralized finance actually decentralized? A social network analysis of the Aave protocol on the Ethereum blockchain. Available from https://doi.org/10.48550/arXiv.2206.0840.

Back, A. et al. (2014). Enabling Blockchain Innovations with Pegged Sidechains.

Belchior, R. et al. (2021). A Survey on Blockchain Interoperability: Past, Present, and Future Trends. Available from https://doi.org/10.48550/arXiv.2005.14282.

Bellavitis, C., Fisch, C. and Momtaz, P.P. (2023). The rise of decentralized autonomous organizations (DAOs): a first empirical glimpse. *Venture Capital*, 25 (2), 187–203. Available from https://doi.org/10.1080/13691066.2022.2116797.

Bhatia, R., Jain, A. and Singh, B. (2021). Hash time locked contract based asset exchange solution for probabilistic public blockchains.

Cardoso, A.G. (2023). Decentralized Autonomous Organizations – DAOs: The Convergence of Technology, Law, Governance, and Behavioral Economics. *SSRN Electronic Journal*. Available from https://doi.org/10.2139/ssrn.4341884.

Chen, L. et al. (2023). Three-Stage Cross-Chain Protocol Based on Notary Group. Available from https://doi.org/10.3390/electronics12132804.

Eisermann, T. et al. (2025). Concentration in Governance Control Across Decentralised Finance Protocols. Available from https://doi.org/10.48550/arXiv.2501.13377.

Falk, B. et al. (2024). Blockchain Governance: An Empirical Analysis of User Engagement on DAOs. Available from https://doi.org/10.48550/arXiv.2407.10945.

Fan, X., Chai, Q. and Zhong, Z. (2020). MULTAV: A Multi-chain Token Backed Voting Framework for Decentralized Blockchain Governance. In: Chen, Z. Cui, L. Palanisamy, B. et al. (eds.). *Blockchain – ICBC 2020*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 33–47. Available from https://doi.org/10.1007/978-3-030-59638-5_3.

Feichtinger, R. et al. (2023). The Hidden Shortcomings of (D)AOs -- An Empirical Study of On-Chain Governance. Available from https://doi.org/10.48550/arXiv.2302.12125.

Frangella, E. and Herskind, L. (2022). Aave V3 Technical Paper. Available from https://github.com/aave/aave-v3-core/blob/master/techpaper/Aave_V3_Technical_Paper.pdf.

Fritsch, R., Müller, M. and Wattenhofer, R. (2024). Analyzing voting power in decentralized governance: Who controls DAOs? *Blockchain: Research and Applications*, 5 (3), 100208. Available from https://doi.org/10.1016/j.bcra.2024.100208.

Gec, S. et al. (2023). A Recommender System for Robust Smart Contract Template Classification. *Sensors*, 23 (2), 639. Available from https://doi.org/10.3390/s23020639.

Guo, H. et al. (2024). A framework for efficient cross-chain token transfers in blockchain networks. *Journal of King Saud University - Computer and Information Sciences*, 36 (2), 101968. Available from https://doi.org/10.1016/j.jksuci.2024.101968.

Hamid Ekal, H. and Abdul-wahab, S.N. (2022). DeFi Governance and Decision-Making on Blockchain. *Mesopotamian Journal of Computer Science*, 9–16. Available from https://doi.org/10.58496/MJCSC/2022/003.

Hardjono, T., Lipton, A. and Pentland, A. (2018). Towards a Design Philosophy for Interoperable Blockchain Systems. Available from http://arxiv.org/abs/1805.05934.

Liu, H. et al. (2024). Using My Functions Should Follow My Checks: Understanding and Detecting Insecure OpenZeppelin Code in Smart Contracts.

Liu, Y. et al. (2023). A systematic literature review on blockchain governance. *Journal of Systems and Software*, 197, 111576. Available from https://doi.org/10.1016/j.jss.2022.111576.

Marella, V. et al. (2020). Understanding the creation of trust in cryptocurrencies: the case of Bitcoin. *Electronic Markets*, 30 (2), 259–271. Available from https://doi.org/10.1007/s12525-019-00392-5.

Messias, J. et al. (2024). Understanding Blockchain Governance: Analyzing Decentralized Voting to Amend DeFi Smart Contracts. Available from https://doi.org/10.48550/arXiv.2305.17655.

Mohammed Abdul, S.S., Shrestha, A. and Yong, J. (2024). CrossDeFi: A Novel Cross-Chain Communication Protocol. *Future Internet*, 16 (9), 314. Available from https://doi.org/10.3390/fi16090314.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. (2008).

Ou, W. et al. (2022). An overview on cross-chain: Mechanism, platforms, challenges and advances. *Computer Networks*, 218, 109378. Available from https://doi.org/10.1016/j.comnet.2022.109378.

Pierro, G.A. and Tonelli, R. (2021). Analysis of Source Code Duplication in Ethreum Smart Contracts. *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. March 2021. Honolulu, HI, USA: IEEE, 701–707. Available from https://doi.org/10.1109/SANER50967.2021.00089.

Rikken, O., Janssen, M. and Kwee, Z. (2023). Governance impacts of blockchain-based decentralized autonomous organizations: an empirical analysis. *Policy Design and Practice*, 6 (4), 465–487. Available from https://doi.org/10.1080/25741292.2023.2270220.

Saim, M. et al. (2022). E-Voting via Upgradable Smart Contracts on Blockchain. *2022 International Conference on Futuristic Technologies (INCOFT)*. 25 November 2022. Belgaum, India: IEEE, 1–6. Available from https://doi.org/10.1109/INCOFT55651.2022.10094482.

Saurabh, K., Rani, N. and Upadhyay, P. (2024). Towards novel blockchain decentralised autonomous organisation (DAO) led corporate governance framework. *Technological Forecasting and Social Change*, 204, 123417. Available from https://doi.org/10.1016/j.techfore.2024.123417.

Shah, N. (2024). Hybrid-DAOs: Enhancing Governance, Scalability, and Compliance in Decentralized Systems. Available from https://doi.org/10.48550/arXiv.2410.21593.

Sion, S.I. et al. (2024). A Comprehensive Review of Multi-chain Architecture for Blockchain Integration in Organizations. In: Di Ciccio, C. Fdhila, W. Agostinelli, S. et al. (eds.). *Business Process Management: Blockchain, Robotic Process Automation, Central and Eastern European, Educators and Industry Forum*. Lecture Notes in Business Information Processing. Cham: Springer Nature Switzerland, 5–24. Available from https://doi.org/10.1007/978-3-031-70445-1_1.

Tan, J. et al. (2024). Open Problems in DAOs. Available from https://doi.org/10.48550/arXiv.2310.19201.

Valiente, M.-C. and Pavón, J. (2024). Web3-DAO: An ontology for decentralized autonomous organizations. *Journal of Web Semantics*, 82, 100830. Available from https://doi.org/10.1016/j.websem.2024.100830.

Xiong, A. et al. (2022). A notary group-based cross-chain mechanism. *Digital Communications and Networks*, 8 (6), 1059–1067. Available from https://doi.org/10.1016/j.dcan.2022.04.012.

Yadav, V. et al. (2024). DataDAO Club - Revolutionizing Investment Management with On-Chain Governance. *2024 International Conference on Expert Clouds and Applications (ICOECA)*. 18 April 2024. Bengaluru, India: IEEE, 333–340. Available from https://doi.org/10.1109/ICOECA62351.2024.00066.

APPENDIX

Appendix A: Concept Map

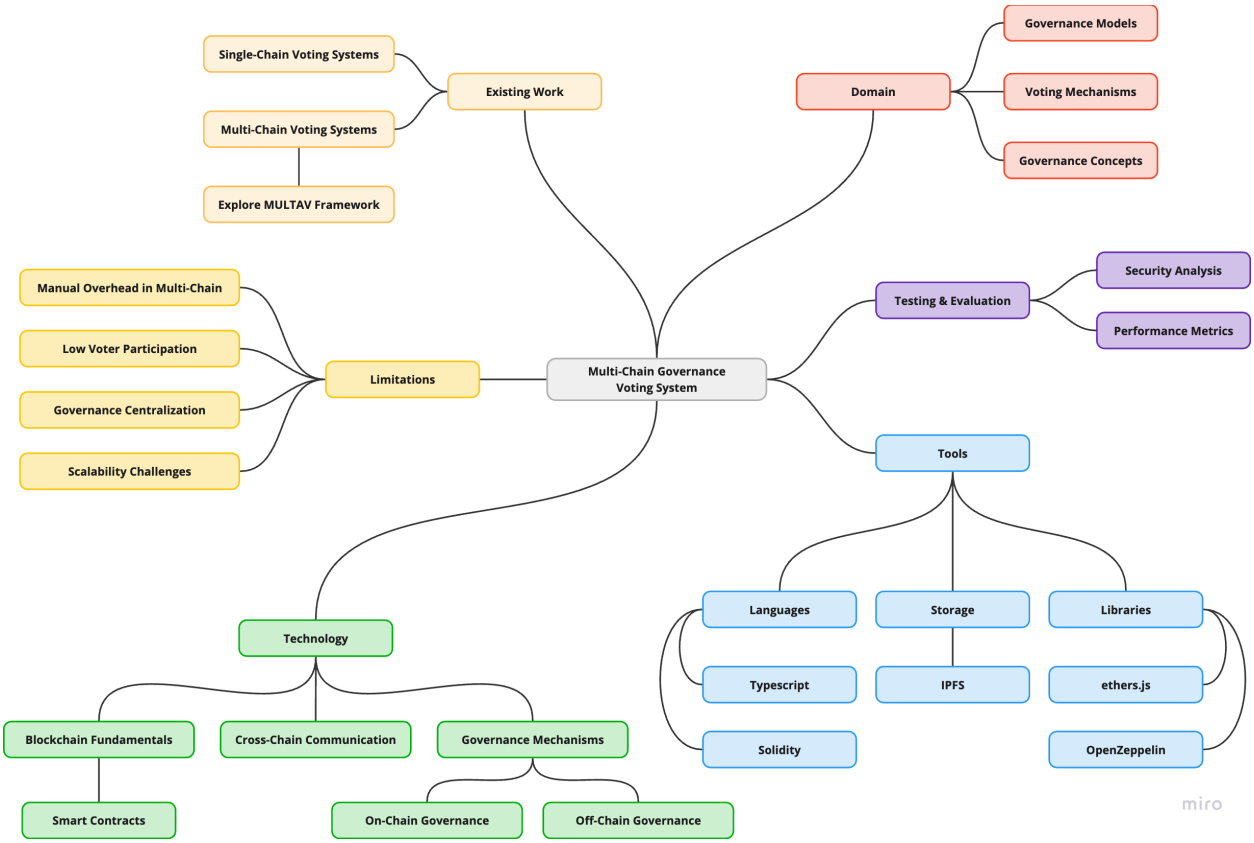


Figure 30: Concept Map (self-composed)

Appendix B: Gantt Chart

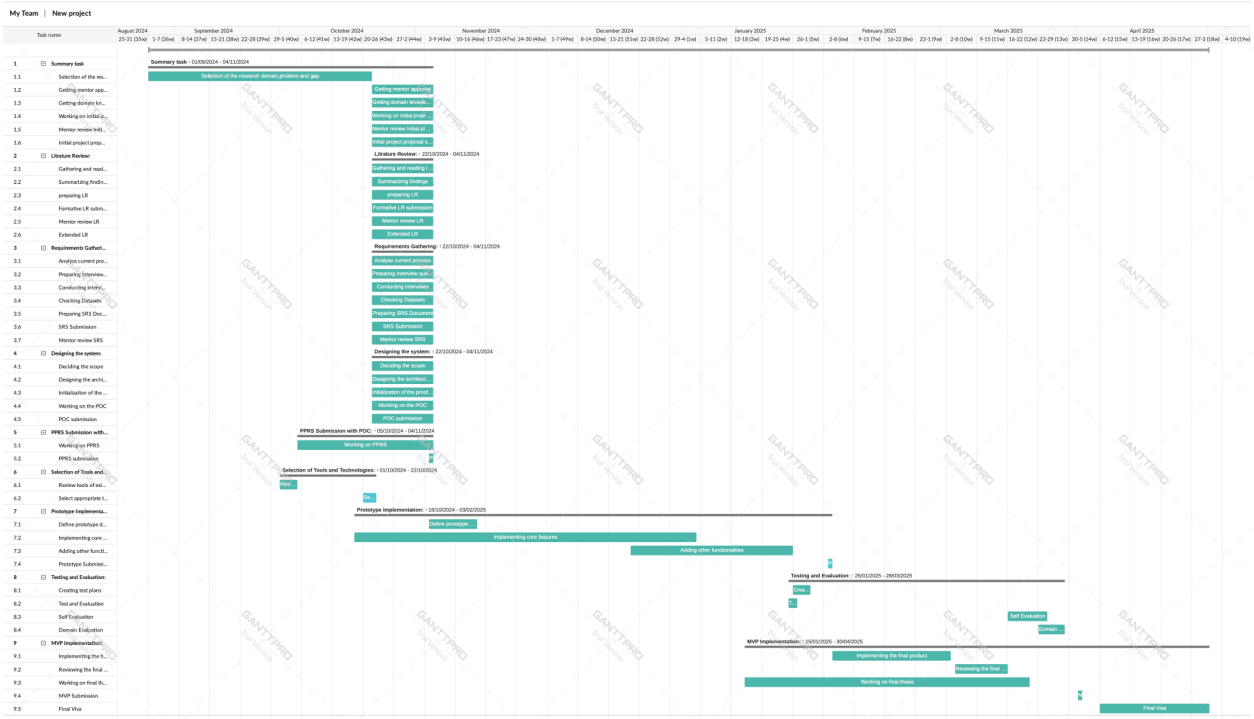
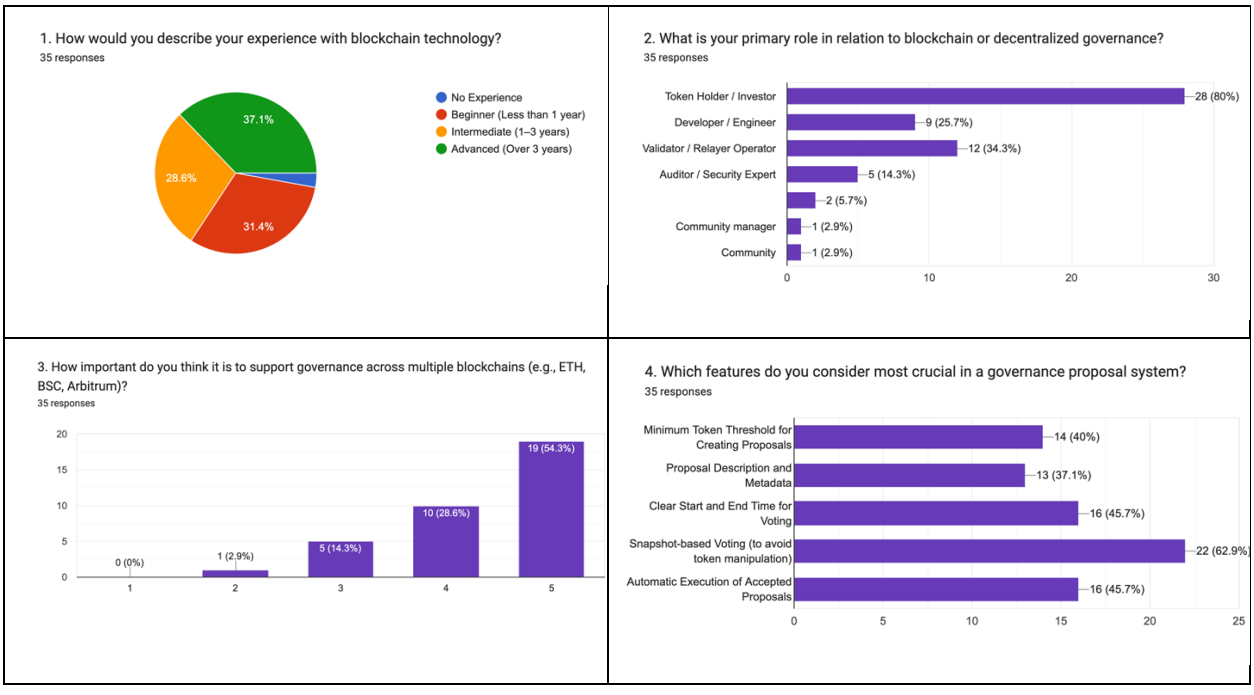


Figure 31: Gantt Chart (self-composed)

Appendix C: Survey Findings



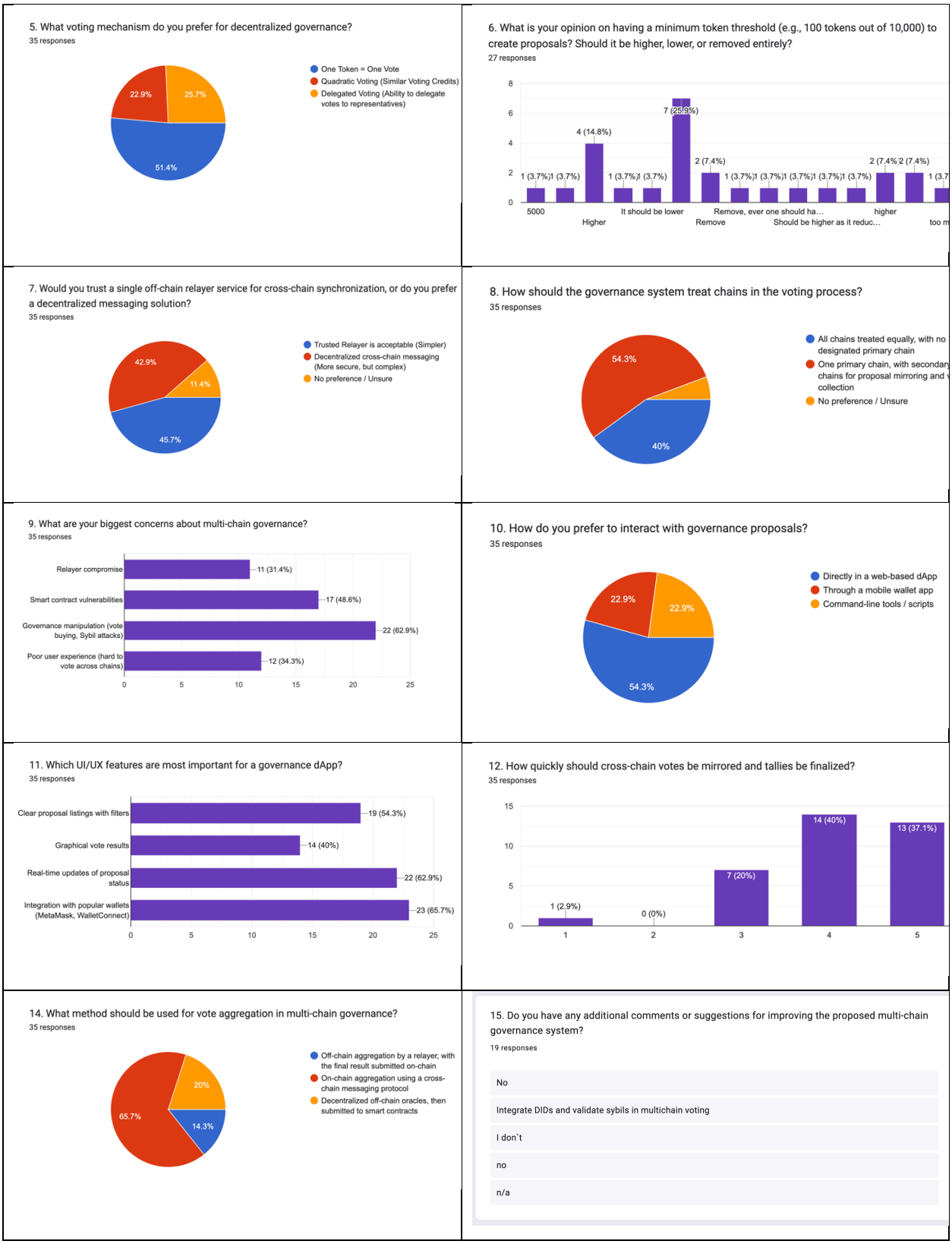


Table 25: Survey findings result

Appendix D: Use Case Descriptions

Use case	Create Proposal
ID	UC1
Actors	User
Description	A user submits a new idea or proposed change for community consideration, initiating the governance process on the main blockchain.
Precondition	The user has sufficient authority or influence within the community to make proposals.
Main flow	1. The user logs into the system. 2. The user provides the necessary information (e.g., title, description, voting duration) 3. The system verifies that the user has the required level of influence. 4. If authorized, the system creates the proposal and sets the voting period. 5. The user receives confirmation that the proposal was submitted successfully.
Alternative flows	A1: Insufficient Influence - At step 3, if the user lacks the required influence, the system informs them and suggests methods to increase their influence, and the process ends without creating a proposal.
Exceptions	- The system is temporarily unavailable due to technical issues. - The user loses internet connectivity during the process.
Postconditions	- A new proposal is successfully created and shared on the main blockchain. - The proposal becomes available for voting within a specified time frame.

Table 26: Use case description for create proposal

Use case	Vote on Proposal
ID	UC2
Actors	User
Description	A user votes on a proposal, either supporting or opposing the suggested change, contributing to the community's collective decision.
Precondition	- An active proposal is available for voting.

	- The user has not previously voted on this proposal in this chain. - The user holds enough influence to vote.
Main flow	1. The user logs into the system. 2. The user selects an active proposal 3. The user chooses to support or oppose the proposal. 4. The system verifies voting eligibility and ensures the proposal is still open. 5. The system confirms the user has voting rights. 6. The system records the vote and updates the tally. 7. The user receives confirmation of a successful vote.
Alternative flows	A1: Voting Period Ended - At step 4, if the proposal is closed, the system informs the user that voting is no longer possible. A2: User Already Voted - At step 4, if the user has already voted, they are notified and cannot vote again. A3: Insufficient Influence - At step 5, if the user lacks influence, the system advises how to gain influence and ends the process.
Exceptions	- The system is unavailable during the voting attempt. - The user loses internet access mid-process.
Postconditions	- The user's vote is recorded. - The chain's total vote count for the proposal is updated.

Table 27: Use case description for vote on proposal

Use case	Finalize Proposal
ID	UC3
Actors	User, Relayer Service
Description	A user or the Relayer Service concludes the voting process, aggregates vote across all blockchains and determines whether the proposal is approved or rejected.
Precondition	- The proposal exists, and its voting period has ended. - A short buffer period has passed to ensure vote readiness. - The final tally has not yet been completed. - For the Relayer Service, all external votes must be available.

Main flow	1. The user or Relayer Service selects a proposal that meets finalization criteria. 2. The system checks eligibility (voting ended, not finalized etc.) 3. A finalization request is made 4. The system collects all relevant votes and total support, and opposition are calculated 5. The outcome is recorded, and the proposal is closed. 6. System will change the status according to the outcome on all chains
Alternative flows	A1: Voting Still Ongoing - At step 2, if the voting period is still active, the process is halted. A2: Proposal Already Finalized - At step 2, if finalization has occurred, the system informs the user. A3: Incomplete External Votes - At step 2, if external votes are unavailable, the process pauses and resumes later.
Exceptions	- System or blockchain access is disrupted. - Technical issues delay vote collection
Postconditions	- All votes are counted. - The proposal is marked as accepted or rejected.

Table 28: Use case description for finalize proposal

Use case	Collect Votes from Secondary Chain
ID	UC5
Actors	Relayer Service
Description	The Relayer Service retrieves vote from secondary blockchains after voting concludes and forwards them to the main blockchain for final tallying.
Precondition	- The voting period on the secondary blockchain has ended. - The votes are available for collection. - They have not yet been sent to the main blockchain. - The Relayer Service is operational.
Main flow	1. The Relayer Service detects that voting has ended and votes are ready. 2. It checks whether the votes have already been collected.

	3. If not, it retrieves the vote totals, and the votes are submitted to the main blockchain 4. The main blockchain records the votes. 5. If all votes are collected, finalization may begin.
Alternative flows	A1: Votes Already Collected - At step 2, if votes were previously submitted, the system proceeds to the next chain A2: Missing Proposal - If the proposal does not exist on the chain, it is mirrored before collecting votes.
Exceptions	- Connectivity issues with the main or secondary blockchain. - System unavailability prevents vote submission.
Postconditions	- The votes are recorded on the main blockchain. - The votes are marked as collected and ready for finalization.

Table 29: Use case description for collect votes from secondary chains

Appendix E: Full-Text Paper Request

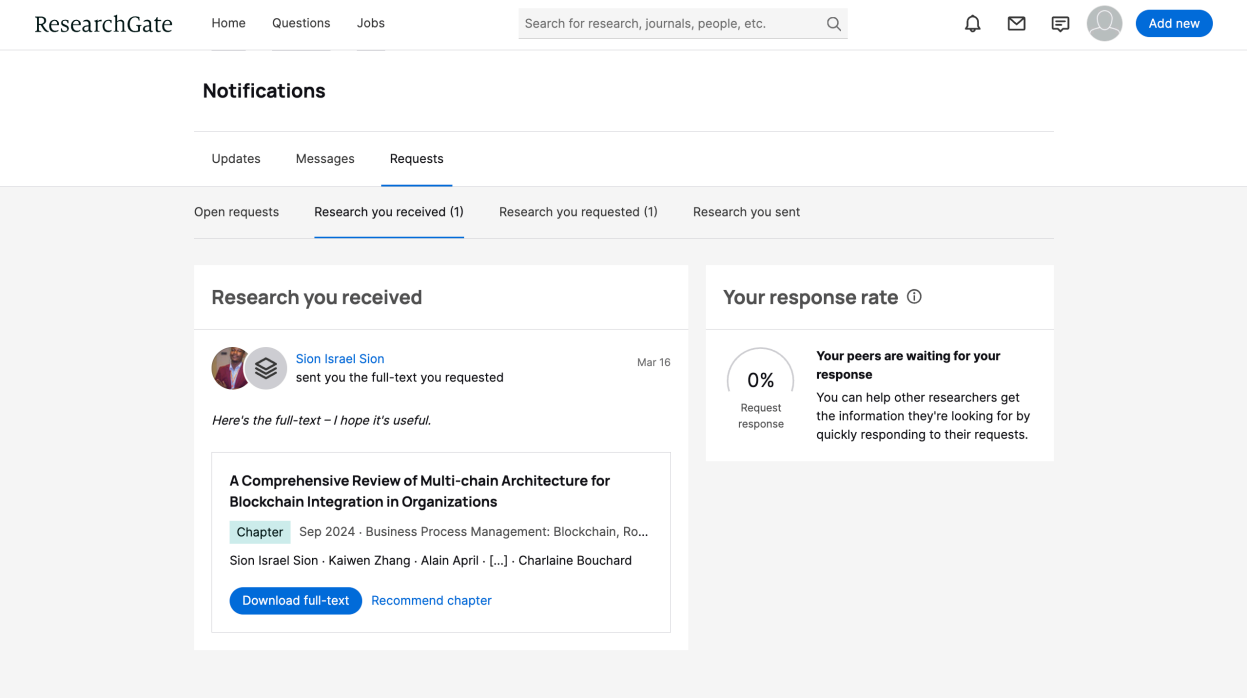


Figure 32: Full text paper request - (Sion et al., 2024)

Appendix F: Low Fidelity UI Design

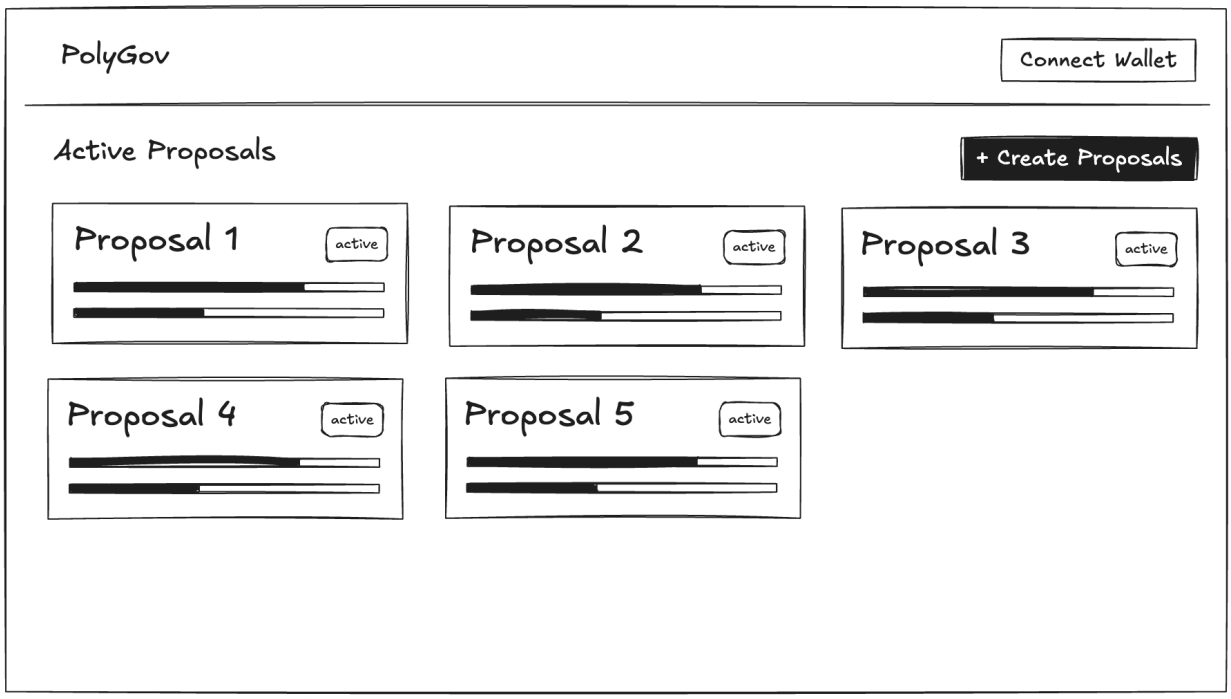


Figure 33: Low Fidelity UI - Proposal List

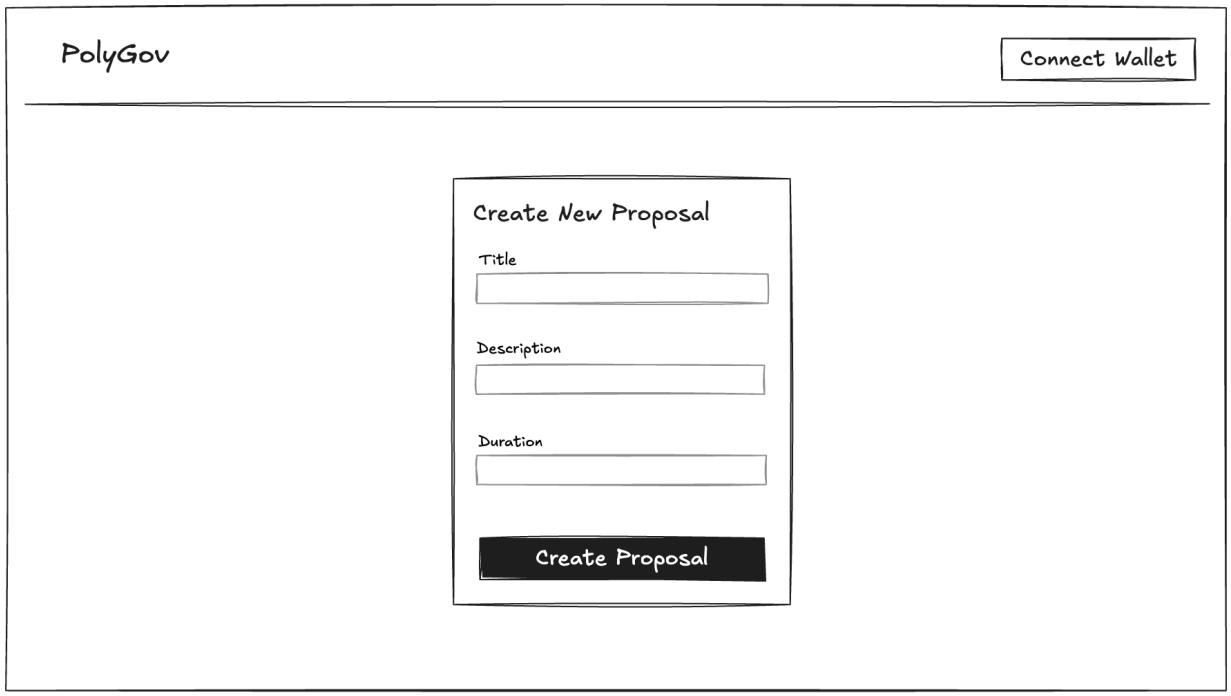


Figure 34: Low Fidelity UI - Proposal Creation Form

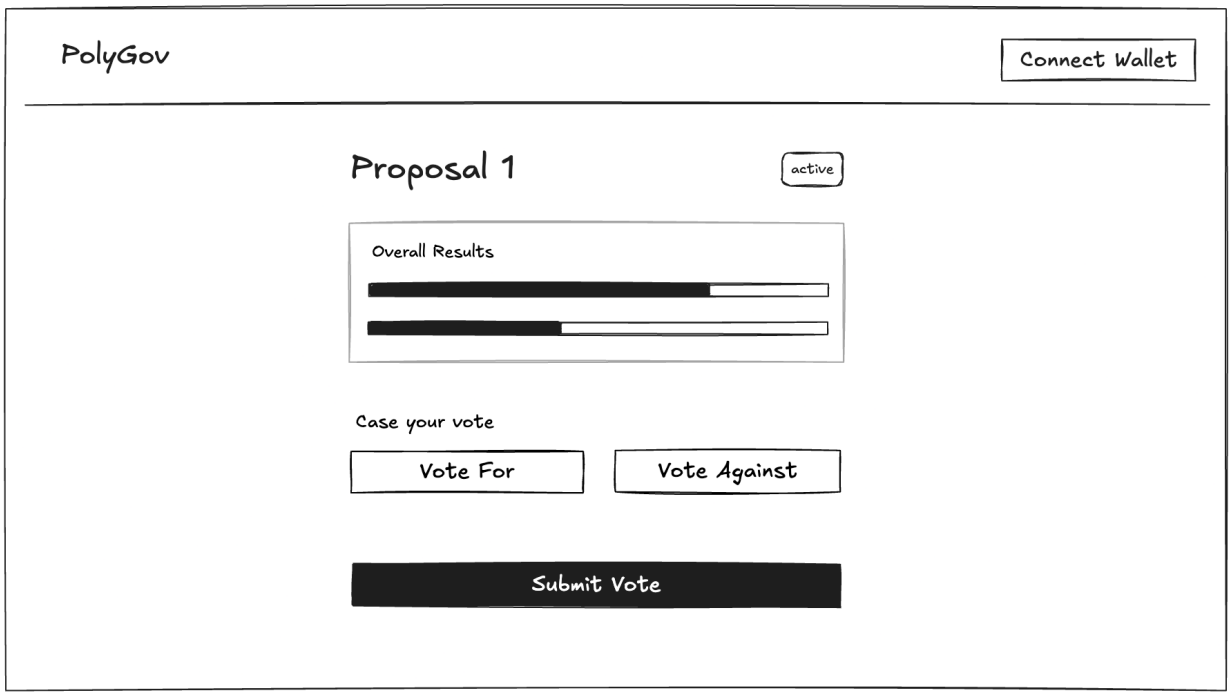


Figure 35: Low Fidelity UI - Proposal Page

Appendix G: High Fidelity UI Design

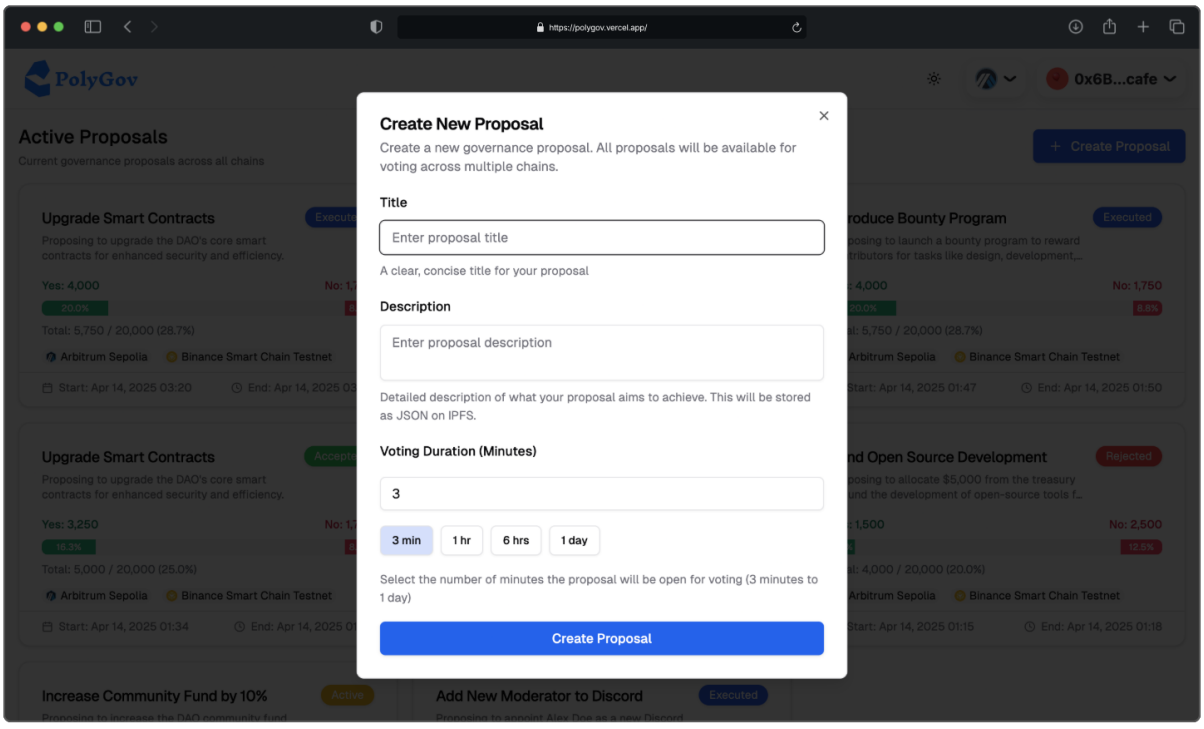


Figure 36: UI - Proposal Creation Form

Appendix H: Performance Testing

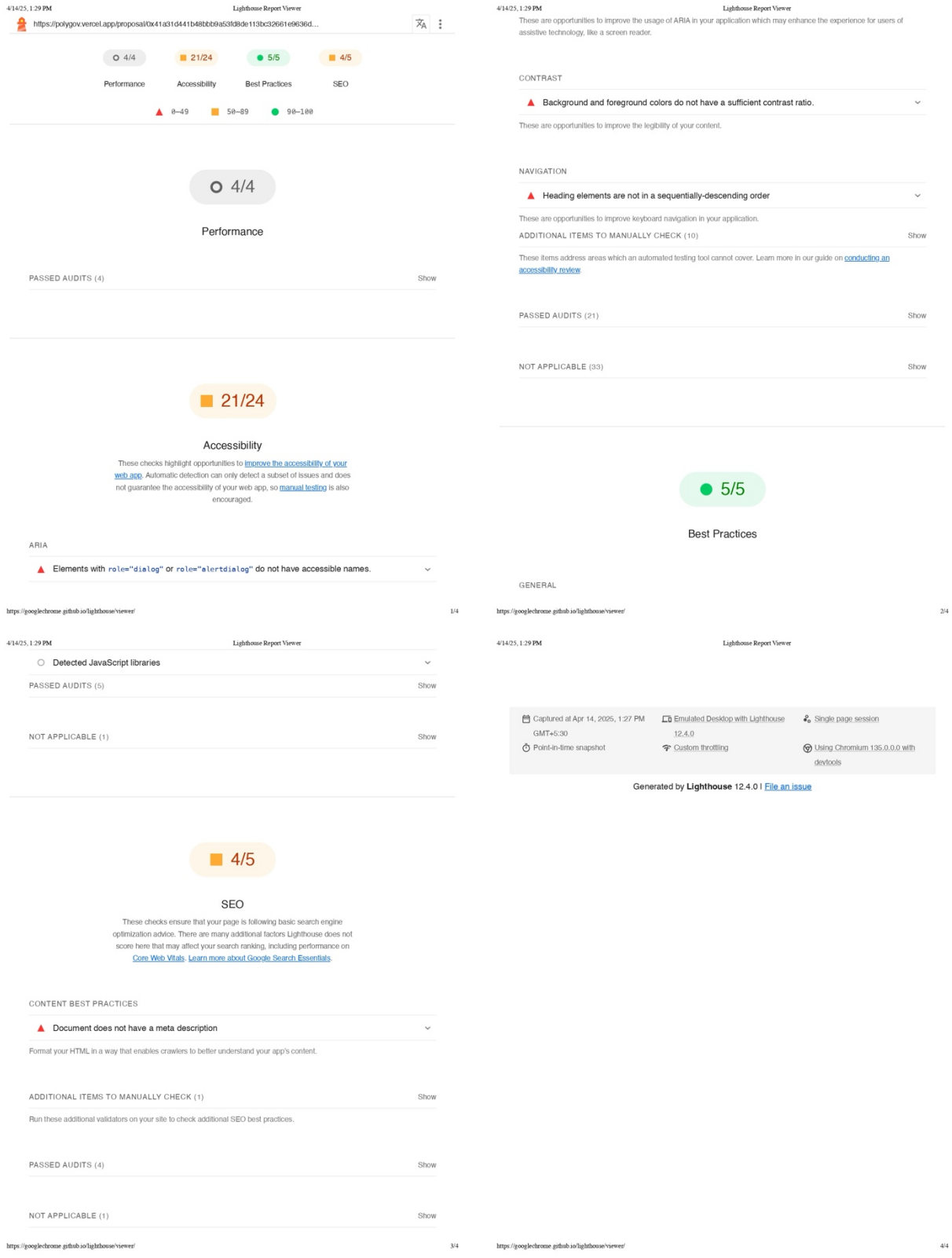


Figure 37: Lighthouse testing report

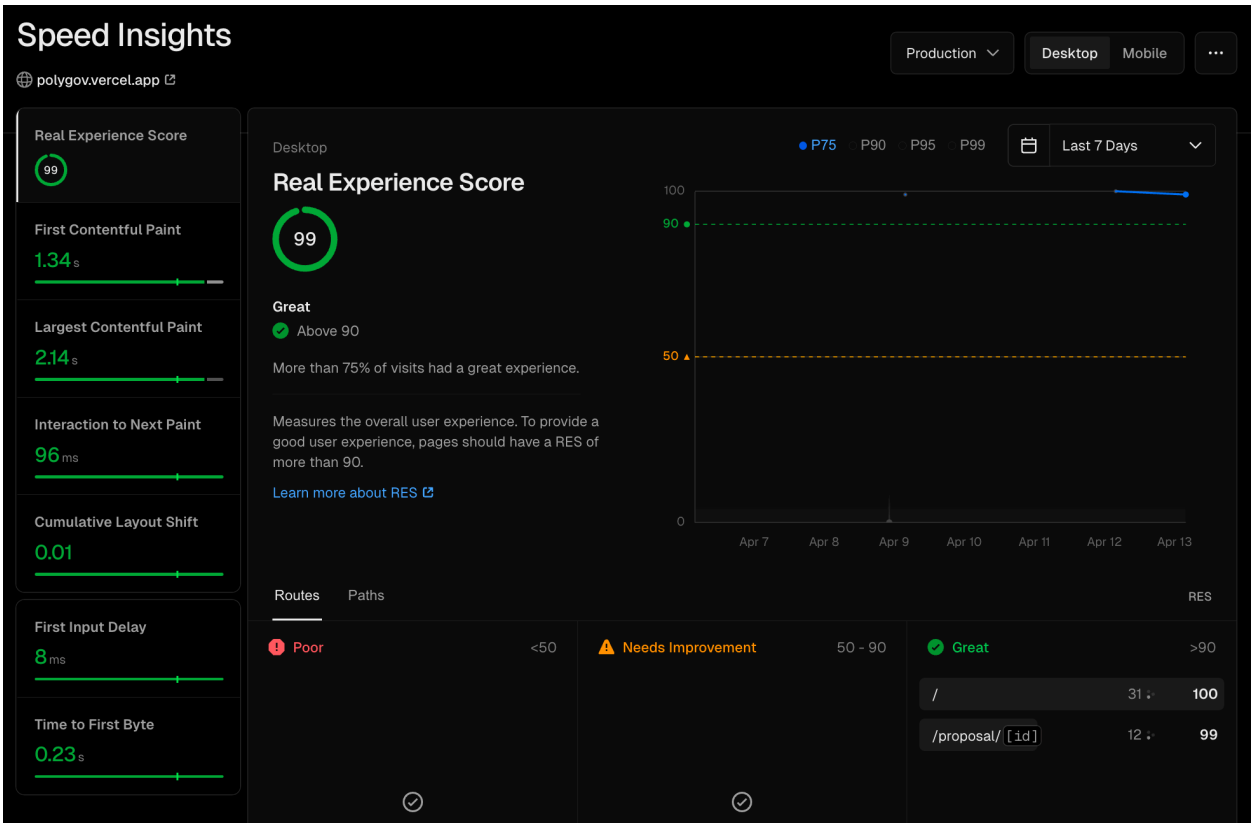


Figure 38: Vercel Speed Insight

Appendix I: Evaluations of Domain & Technical experts

Question1: What do you think about the goal of the PolyGov project?

Evaluator ID	Feedback
EV1	This appears to be a solid solution, will be a crucial system for future DAOs operating across multiple blockchains
EV2	I see that the project is about fixing critical challenges with multi chain governance, this fix will be something highly valuable.
EV3	Solid problem area. Definitely important for the future of DAOs and worth investing time into.
EV4	Yes it is PolyGov is much needed solution for Multichian Governance
EV5	The goal of PolyGov is both timely and impactful. As decentralized ecosystems grow across multiple chains, a unified governance layer is essential. PolyGov

	addresses a real problem by enabling multichain voting, which could enhance inclusivity and decentralization in protocol governance.
EV6	Strong use case. Very useful for cross-chain interoperability for governance.

Table 30: Experts feedback for Question 1

Question2: How would you evaluate the implementation of the project so far?

Evaluator ID	Feedback
EV1	Core features are functional; however, incorporating crosschain execution automation would enhance its completeness
EV2	This implementation is covering core voting and proposal mechanisms at this stage, advanced improvement is necessary to make it fully decentralized.
EV3	Working as expected for a demo. Performance is good for now but will need scaling as the project grows.
EV4	Current implementation covered but it totally covers the EVM side chains and L2s without any issue.
EV5	The use of off-chain relayers to synchronize contracts across chains is a pragmatic approach to achieving cross-chain communication without incurring excessive on-chain gas costs. However, this also introduces trust and liveness assumptions on the relayer, which need careful design, monitoring, and ideally some redundancy or decentralization to avoid being a single point of failure.
EV6	System works smoothly at demo scale. No major issues spotted so far.

Table 31: Experts feedback for Question 2

Question3: What is your opinion about the system design?

Evaluator ID	Feedback
EV1	It appears solid. Simple and straightforward design that's well suited for a proof of concept.
EV2	The infrastructure of this project looking clean and comprehensive for a cross chain governance prototype.

EV3	Decent overall. The relayer being a centralized point is something to keep an eye on as it could become a weakness.
EV4	Much needed requirement to the Multichian ecosystem since there are many chains coming to the market
EV5	The system design is ambitious and forward-thinking. Relying on off-chain relayers is a practical compromise between decentralization and efficiency. It is beneficial to see modular components that allow for plug-and-play relayer providers and customizable governance logic per chain, improving flexibility and adaptability.
EV6	Nice separation of concerns between contracts and the relayer. Clear structure makes it easier to maintain and audit

Table 32: Experts feedback for Question 3

Question4: Do you have any suggestions to improve the system?

Evaluator ID	Feedback
EV1	Try integrating it into more blockchains (Polkadot and other blockchains built with frameworks like the Cosmos SDK)
EV2	Maybe it's better to consider about reducing relayer trust assumptions in the next version of this.
EV3	Try exploring decentralized relayers or multisig options to reduce reliance on a single point and improve resilience
EV4	Enable more non EVM chains and UTXO enabled chains if possible and try to get all votes at once.
EV5	- Explore decentralizing the relayer network, potentially through a mesh of validators or a watcher network to reduce single points of failure. - Introduce formal dispute resolution mechanisms to address potential misbehavior or failures by off-chain relayers. - Improve transparency by implementing verifiable audit logs or traceable event histories for off-chain activities. - Investigate the integration of emerging technologies like LayerZero to enhance decentralization, fault tolerance, and cross-chain availability.

EV6	Prioritize thorough security audits before moving into production to ensure stability and trust

Table 33: Experts feedback for Question 4

Appendix J: Deployed Contracts

Contract Name	Chain	Contract Address	Tx
PGVToken	ARB	0xc359f38eD76d8941d486dA85a4aA553Aa74b18BD	TxHash
	BSC	0xc359f38eD76d8941d486dA85a4aA553Aa74b18BD	TxHash
MainGovernance	ARB	0x2265d043C79CaECA800be7EcC150C6C23D5E3374	TxHash
SecondaryGovernance	BSC	0x216941d5ce326edec59ffdf6959a1b855ee94276	TxHash
DemoContract	ARB	0x54DccD4b6dca0a13767A17899E706911Cdf8D106	TxHash

Table 34: Deployed Contracts

Appendix K: Solana Integration Draft

PolyGov - Sol1 Rust

```rust
use anchor_lang::prelude::*;

declare_id!("6w7GnthMZ46Uf1exG3uaNbwZfcpJDhY4uWuBbXM8sd8s");

#[program]
pub mod multigov_voting {
    use super::*;

    pub fn initialize_config(ctx: Context<InitializeConfig>, relayer: Pubkey) -> Result<()> {
        let config = &mut ctx.accounts.config;
        config.relayer = relayer;
        config.admin = ctx.accounts.admin.key();
        Ok(())
    }

    pub fn set_proposal(ctx: Context<SetProposal>, proposal_id: u64, end_time: i64) ->
Result<()> {
        let proposal = &mut ctx.accounts.proposal;
        proposal.id = proposal_id;
        proposal.end_time = end_time;
        Ok(())
    }

    pub fn vote(ctx: Context<Vote>, proposal_id: u64, support: bool) -> Result<()> {
        let proposal = &mut ctx.accounts.proposal;
        let voter = &mut ctx.accounts.voter;

        require!(proposal.id == proposal_id, ErrorCode::InvalidProposal);
        require!(Clock::get()?.unix_timestamp < proposal.end_time, ErrorCode::VotingEnded);
        require!(!voter.has_voted, ErrorCode::AlreadyVoted);

        if support {
            proposal.yes_votes += 1;
        } else {
            proposal.no_votes += 1;
        }

        voter.has_voted = true;
        Ok(())
    }

    pub fn update_relayer(ctx: Context<UpdateRelayer>, new_relayer: Pubkey) -> Result<()> {
        let config = &mut ctx.accounts.config;
        config.relayer = new_relayer;
        Ok(())
    }
}

#[derive(Accounts)]
pub struct InitializeConfig<'info> {
    #[account(
        init,
        payer = admin,
        space = 8 + 32 + 32,
        seeds = [b"config"],
        bump
    )]
    pub config: Account<'info, Config>,

    #[account(mut)]
    pub admin: Signer<'info>,

    pub system_program: Program<'info, System>,
}

#[derive(Accounts)]
#[instruction(proposal_id: u64)]
pub struct SetProposal<'info> {
    #[account(
        init_if_needed,
        payer = relayer,
        space = 8 + 8 + 8 + 8 + 8,
        seeds = [b"proposal", proposal_id.to_le_bytes().as_ref()],
        bump
    )]
    pub proposal: Account<'info, Proposal>,

    #[account(
        seeds = [b"config"],
        bump
    )]
    pub config: Account<'info, Config>,

    #[account(mut, signer, constraint = relayer.key() == config.relayer @
ErrorCode::UnauthorizedRelayer)]
    pub relayer: Signer<'info>,

    pub system_program: Program<'info, System>,
}
```

PolyGov - Sol2 Rust

```rust
#[derive(Accounts)]
#[instruction(proposal_id: u64)]
pub struct Vote<'info> {
    #[account(
        mut,
        seeds = [b"proposal", proposal_id.to_le_bytes().as_ref()],
        bump
    )]
    pub proposal: Account<'info, Proposal>,

    #[account(
        init_if_needed,
        payer = user,
        space = 8 + 1,
        seeds = [b"voter", user.key().as_ref(), proposal.key().as_ref()],
        bump
    )]
    pub voter: Account<'info, Voter>,

    #[account(mut)]
    pub user: Signer<'info>,

    pub system_program: Program<'info, System>,
}

#[derive(Accounts)]
pub struct UpdateRelayer<'info> {
    #[account(
        mut,
        seeds = [b"config"],
        bump
    )]
    pub config: Account<'info, Config>,

    #[account(mut, signer, constraint = admin.key() == config.admin @
ErrorCode::UnauthorizedAdmin)]
    pub admin: Signer<'info>,

    pub system_program: Program<'info, System>,
}

#[account]
pub struct Config {
    pub relayer: Pubkey,
    pub admin: Pubkey,
}

#[account]
pub struct Proposal {
    pub id: u64,
    pub yes_votes: u64,
    pub no_votes: u64,
    pub end_time: i64,
}

#[account]
pub struct Voter {
    pub has_voted: bool,
}

#[error_code]
pub enum ErrorCode {
    #[msg("Voting period has ended")]
    VotingEnded,
    #[msg("User has already voted")]
    AlreadyVoted,
    #[msg("Invalid proposal ID")]
    InvalidProposal,
    #[msg("Unauthorized: Only the relayer can set proposals")]
    UnauthorizedRelayer,
    #[msg("Unauthorized: Only the admin can update the relayer")]
    UnauthorizedAdmin,
}
```

Figure 39: Deprecated Solana Integration

Appendix L: Sepolia Issue

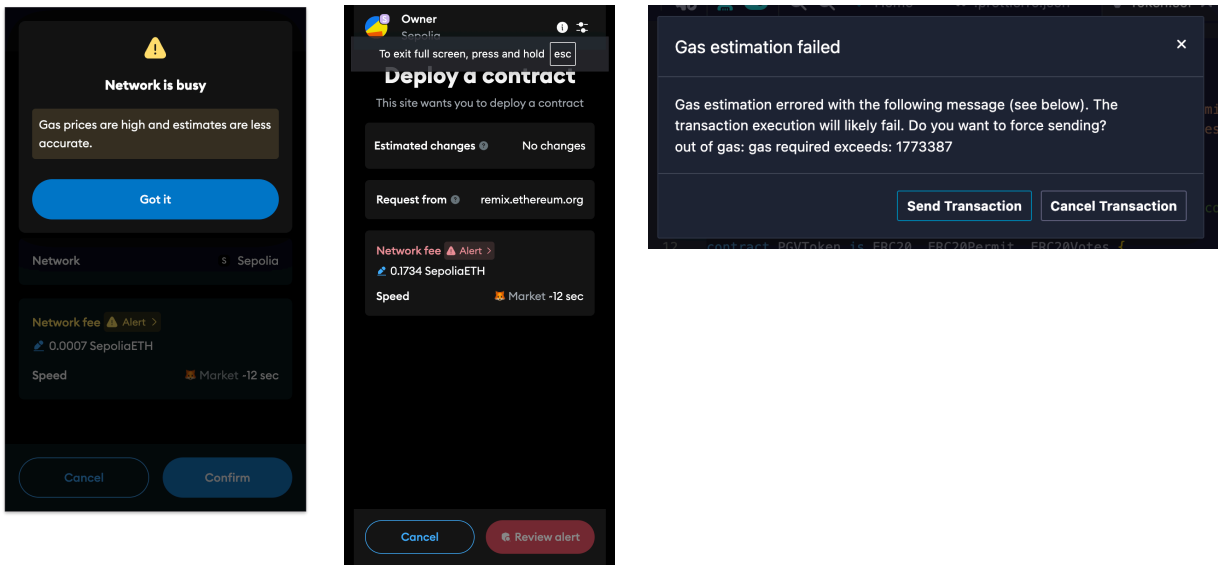


Figure 40: Sepolia Gas Fee Estimation Issue on Remix and MetaMask