A Comprehensive Framework for Evaluating Different Layer 2 Protocols in Ethereum

by

Md. Mahadi Hassan Riyadh
22241157
Md. Safwoan Saliken
21301076
Wasima Annan
21201362
Arjun Ghosh
21201572
Eshat Fareeha
21201619

A **thesis defense report** submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
June 2025

Declaration

It is hereby declared that

1. The report of thesis defense submitted is our own original work while completing degree at Brac University.
2. The report does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The report does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Mahadi Hassan Riyadh

22241157

Md. Safwoan Saliken

21301076

Wasima Annan

21201362

Arjun Ghosh

21201572

Eshat Fareeha

21201619

Approval

The thesis defense report titled "A Comprehensive Framework for Evaluating Different Layer 2 Protocols in Ethereum" submitted by

1. Md. Mahadi Hassan Riyadh (22241157)
2. Md. Safwoan Saliken (21301076)
3. Wasima Annan (21201362)
4. Arjun Ghosh (21201572)
5. Eshat Fareeha (21201619)

Of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 20, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Md Sadek Ferdous, PhD

Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Mr. Rafeed Rahman

Senior Lecturer
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Associate Professor; Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Blockchain technology was created with the motive to establish a decentralized, secured, and transparent system for digital transactions, reducing fraudulent behavior and aiming to stabilize the economy. However, scalability has always been regarded as a key issue that has been limiting the capacity of the system. With the widespread use of blockchain systems across the world, the demand for a high-speed transaction and lower latency in finality has increased. This is a significant barrier with the growing chain as it takes a substantial amount of time to process the transaction in Layer 1 which also adds in additional cost that makes each transaction more expensive and less user friendly. One of the leading solutions to scalability issues has been rollups that conduct transactions outside of the main chain and publish the summary of transactions in the main chain. However, rollup technology is relatively new and the protocols built on top of this are not very user-friendly. In this research, we propose and develop a Layer 2 recommendation system based on the user's needs. The system is capable of giving users score-based feedback for different Layer 2 protocols. It can also deploy the user's smart contract on the test networks of one or more of these Layer 2 protocols if the user wishes to do so. The system is aimed to reduce the hassle of users, promoting increased use of Layer 2 protocols. In this research, we introduce **L2Easy**, a Layer 2 protocol recommendation system that is customized to individual user needs. The system provides score-based evaluations of various Layer 2 protocols based on customizable criteria and offers the functionality to deploy user defined smart contracts to test networks across selected protocols. Our goal is to streamline the decision making process for users, increase the usability of the protocol, and promote a wider adoption of Layer 2 technologies.

Index Terms: Blockchain, Ethereum, Layer 2, Rollups, Smart Contract, Protocol, Comparison, Evaluation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent times, blockchain has gained tremendous popularity in the industrial, economic, and academic setting. With Bitcoin trailblazing the industry of cryptocurrency and initiating the genesis of the first generation of blockchain [15], has inspired the creation of numerous innovative concepts and platforms to promote digital tokens, decentralization, and cyber security. Blockchain technology has evolved significantly and is now being used in developing decentralized applications (dApps) and smart contracts, which brings us to the second generation of blockchains. Ethereum, being the most favored public blockchain for dApp development and smart contracts, has opened doors to a plethora of utilities and use cases for blockchain. In spite of the many advantages of using this technology, the globalization of blockchain is still hindered as a consequence of some limitations, the most significant one being scalability issues due to high transaction fees and delayed transaction time.

One of the major bottlenecks of blockchain technology is the blockchain trilemma. The trilemma states that it is difficult to achieve security, decentralization, and scalability of a blockchain system simultaneously. It dictates that one of these aspects must be compromised in order to achieve the other two. Due to the limited usage of early blockchain systems, developers often sacrificed scalability to maximize security and decentralization. However, with time the increasing popularity of blockchain systems not only increased its user base but also the complexity of applications built on top of these systems. As Ethereum was the first Turing complete blockchain platform, it lacks scalability in order to achieve other two properties. However, the large number of users as well as smart contracts requires completion of large amounts of transactions each second on Ethereum main-net. But due to the scalability bottleneck, Ethereum main-net can only complete a small portion of required transactions per second. As a result, the platform is not usable in many real life scenarios.

Many solutions have been created in order to solve the scalability issue and make Ethereum more accessible to users and developers. To resolve these difficulties, Layer 1 solutions were first proposed, which focused on modifying the consensus algorithm and the block data; and network sharding [18]. While Layer 1 helped make Ethereum more scalable, it was still not quite practical for the existing blockchains, which led the path to Layer 2 solutions to make the Ethereum blockchain more scalable and

also to reduce the load on Layer 1. Among these solutions, the layered architecture or Layer 2 model shows high potential in improving the scalability of the platform. Layer 2 architecture moves and performs most of the transaction on a chain outside the main chain and only updates a final transaction summary in the main chain. As a result, the Ethereum mainnet has to process much less transactions than the actual number of executed transactions. This approach improves the scalability issue of Ethereum while utilizing the security and decentralization of Ethereum. Thus, making it more appealing for mass users and developers. Amongst various Layer 2 protocols, optimistic rollups and zero-knowledge proofs are widely used. Arbitrum and Optimism are popular projects that utilize optimistic rollups and have gained a lot of popularity recently. On the other hand, ZKsync Era and Starknet are zero-knowledge proof based projects that show huge future potential. These mechanisms allow developers to build more complex applications on Ethereum platform and use them in more real life scenarios. As a result, participation of users and developers in the Ethereum ecosystem is increasing rapidly.

To address the challenges regarding usability and accessibility of Layer 2 solutions, we introduce **L2Easy**, a framework designed to help users choose the most suitable Layer 2 protocol based on their preference. This framework evaluates some leading protocols - Arbitrum, Optimism, and ZKsync Era - using a score-based approach by taking in priorities defined by the user on the basis of scalability, security, decentralization, cost efficiency, and developer experience. Other than comparative evaluation, L2Easy lets users deploy their smart contract on the test networks of the chosen protocols through an intuitive interface. Our system simplifies decision-making, ensures seamless smart contract deployment, and helps promote the broader adoption of Layer 2 solutions in real world applications.

1.1 Research Objectives (RO)

- **RO1:** To critically analyze the current state-of-the-art research and systems within the scope of Ethereum Layer 2 solutions, focusing on scalability, security, and interoperability for Ethereum-based applications.
- **RO2:** To perform user preference, heuristic and security analysis of selected Layer 2 protocols.
- **RO3:** To design an architecture for the proposed Layer 2 evaluation framework that meets all defined functional and non-functional requirements, including performance, security, and interoperability.
- **RO4:** To propose a robust framework and implement a system to evaluate the feasibility and usability of our proposed framework.

1.2 Structure

The report is organized as follows: Chapter 1 introduces the work and outlines the research objectives, detailing the aims, scope, and guiding questions of the project. Chapter 2 presents the background and terminologies related to blockchain and Layer 2 solutions, providing foundational knowledge for the rest of the paper. Chapter 3 discusses about an in depth review of relevant literature and existing work in the field while covering both technical documentation and academic research related to existing Layer 2 protocols. Chapter 4 offers an outlook of the proposed system which contains the methodology, system proposal, analysis od the core requirements and a thorough discussion on the proposed framework. The system's architecture and protocol flow, detailing the technical design and evaluation criteria used to assess four prominent Layer 2 solutions - Arbitrum, Optimism, Starknet, and ZKSync Era are introduced in Chapter 5. Chapter 6 contains the implementation of the proposed system, including both frontend and backend components, the evaluation engine and it's pseudocode, deployment engine, and compilation engine. Finally, Chapter 7 offers a discussion on requirement fulfillment, research objective analysis, encountered limitations, and future development plans, followed by the conclusion in Chapter 8.

Chapter 2

Background

This section is mainly for those less familiar with Layer 2 or blockchain in general. This section will give you an idea about the blockchain space. We start the section with the background of blockchain and categorize it into two types, after which we discuss Ethereum. We briefly explain different blockchain systems and their limitations. Finally, we talk about Ethereum Layer 2 and briefly mention some of the protocols. Details of each of these protocols are discussed in a later part of the paper in Section 3.1.

2.1 Blockchain

The main reason why blockchain came into existence was due to the lack of trust in single authorities [7]. When one party controls the data stored on a server, the data in such a centralized system has a risk of being manipulated by the owner. The solution to this problem is decentralization. Here, rather than one party having full control, decisions are made collectively, which is verified by all participants [20]. This idea of decentralization is the primary concept in blockchain. Blockchain is a decentralized, distributed ledger system which consists of consecutive blocks chained together [13]. It uses a peer-to-peer network for synchronization of data, where all participants use private or public keys to interact with the network. The blocks follow a set of rules, and each of the blocks is created at a predefined interval, or after a certain event occurs, in a decentralized manner. Blockchain systems depend on a consensus mechanism to ensure that distributed nodes agree on the state of particular data [14]. This validation process takes place with the help of consensus algorithms such as Proof of Work (PoW), Proof of Stake (PoS), or Delegated Proof of Stake (DPoS). Within each block, data is stored, and the integrity of data is maintained using cryptographic hash. Once data is recorded in a blockchain system, it becomes nearly impossible to change it, which provides security.

Satoshi Nakamoto introduced Bitcoin in 2009 [1]. Bitcoin is the first decentralized currency based on blockchain. Here, digital currency can be transferred without relying on third parties such as banks. Although Bitcoin focuses only on digital

currencies, in 2015, Vitalik Buterin introduced Ethereum, where the development of decentralized applications (dApps) and the creation of smart contracts are possible. This allows Ethereum to have more potential than Bitcoin. While Bitcoin only offers a single application of blockchain technology, Ethereum, in contrast, allows for a much broader range of uses by running dApps using smart contracts [4]. Due to Ethereum's adaptability, in our research, we will be moving forward with Ethereum and its Layer 2 solutions.

2.2 Types of Blockchain

Blockchain is mainly categorized into two types: public and private [11] as discussed below:

- **Public Blockchain:** Public blockchains are also known as permissionless blockchains. Here, anyone can participate in the blockchain system, and no single entity has control over the network. Decisions are made by consensus among participants. Any users can participate in the decision-making process, and may or may not get rewarded for their participation [5]. As a result, there is no trusted third party or central ledger in a public blockchain [3]. The blockchain state and its data are transparent to everyone, which means transactions are also visible to everyone. However, participants can remain pseudo-anonymous. This may raise privacy concerns where the privacy of certain data needs to be preserved. However tampering with public blockchains is almost impossible, as public blockchains store transactions in different nodes in the distributed network [6]. Public blockchains are the foundation for cryptocurrencies like Bitcoin and Ethereum.
- **Private Blockchain:** Private blockchains are also known as permissioned blockchain. Here, access to the network is restricted, and only the authorized entities can participate in the activities within the blockchain. Hence the privacy concerns mentioned in public blockchain can be solved with private blockchain. However, this is a more centralized approach. Hyperledger [24] and Corda [27] are examples of private blockchain.

2.3 Ethereum

Ethereum [60] is a decentralized, open-source blockchain platform with smart contract functionality. Here, developers can build and deploy smart contracts decentralized applications (dApps). Ethereum introduces a Turing-complete virtual machine, called Ethereum Virtual Machine (EVM). With the help of the EVM, smart contracts, which are computer programs, are executed without the help of any intermediary. The smart contracts are executed using transactions and they ultimately change the state of the EVM. The network is powered by Ethereum's native cryptocurrency, Ether (ETH), which rewards miners and pays transaction fees, known as

gas. Ethereum can be thought of as a World Computer since it acts as a distributed virtual computer where applications can be developed. Due to Ethereum being programmable and flexible, it has become a foundation for non-fungible tokens (NFTs), decentralized finance (DeFi), and other blockchain-based applications.

2.4 Limitations of Base Layer Chains

Even though blockchain technology has introduced numerous innovations, it faces a huge number of limitations that do not allow it to be used to its full potential. While enabling secure, transparent, and tamper-resistant digital ledgers, blockchain offers a new paradigm of trustless systems where intermediaries are no longer necessary for validating and executing transactions; the limitations reduce the efficiency of the entire system. This reduces the widespread usage of blockchain. These limitations include scalability issues, blockchain bloating, blockchain trilemma, security concerns, interoperability issues, power consumption and associated costs, etc. Introducing Layer 2 solutions is a way to reduce these limitations. Layer 2 interacts with the blockchain network in two ways: assets moving from the main chain and vice versa. In other words, locking the assets on the main chain as the transaction carried out in Layer 2 [26]. Those limitations are as follows:

- **Scalability Issues:** One of the most significant challenges in blockchain technology is scalability. A blockchain network can handle a limited number of transactions per second (TPS). For instance, Bitcoin can handle approximately 3-7 TPS [38] while Ethereum handles around 15 TPS [26]. Even though the maximum theoretical TPS of bitcoin is 12.3, in June 2019, Blockchain.com had the average TPS of 3.88 with 2,328 transactions per block [38]. Which is very low if it is compared with the theoretical maximum. Even blocks have size limitations. On average, Ethereum can include approximately 200 transactions per block [26]. If block size is increased then TPS also increases, which takes more time to propagate the block among the chain, which reduces the scalability.

- **Blockchain Bloating:** The issue of blockchain bloating is related to the underlying architecture of blockchain itself [31]. As days go by, the size of blockchain is increasing. For example Bitcoin size is currently more than 500 GB and still increasing while Ethereum size is currently more than 1 TB. This bloating of the blockchain increases the time of processing each transaction. This data pileup creates a domino effect as it can lead to a number of problems such as storage and bandwidth issues as it is needed to keep dispersed nodes in sync [31].

- **Blockchain Trilemma:** A trilemma is a scenario where it can be achieved to get any 2 out of 3 outcomes. The blockchain trilemma which was first mentioned by Vitalik Buterin is a trade-off or a situation where two possible properties can be achieved among the three. The properties are security, scalability and decentralization. Among these three properties only two of them

can be achieved at once but decentralization is the key property and security is a necessity so for most cases the trade-off happens in case of scalability [12]. The key challenge is to achieve all three of these properties all together without any trade-off. This is the infamous blockchain trilemma.

- **Security Issues:** A blockchain system is prone to many attacks like DDoS attack, 51% attack, sybil attack, selfish mining and many more. Most of these attacks can be curated by malicious nodes or validators who disguise themselves as honest nodes but in reality they work as malicious nodes to fulfill personal or group malicious agenda which results in loss of the user and illegal gain of those validators. For example, the leading issue in Polygon with the proceedings of validation is primarily the 51% attack which occurs when more than half of the validator nodes are controlled by one or a group of malicious nodes [16]. The lack of proper security in Layer 2 will lead to reducing the usage of the system.

- **Interoperability Issues:** In the blockchain systems, cross chain communications are still limited. The process of transferring data or assets among various blockchain is still not as fluent and has limitations. Which can be solved by using Layer 2 solutions.

- **Power Consumption and Associated Cost:** Public blockchains which use PoW as their consensus algorithm utilizes more computational power than the ones using PoS. More computational power means more electricity consumption which takes a huge toll on the environment. And for the additional computational power the associated cost also increases. Even to use Ethereum public blockchain network which mostly uses PoS as their consensus algorithm, a validator needs to have a capital that will be stored in an escrow account, to be able to be a validator. This is also an extra cost that is included to run the system.

2.5 Layer 2 Solutions

Layer 2 solutions represent a critical advancement in the evolution of blockchain technology, specifically designed to address some of the most pressing limitations faced by Layer 1 networks. As the adoption and demand for blockchain-based applications grow, Layer 1 networks have encountered bottlenecks such as limited scalability, high transaction fees, and congestion.

These solutions aim to mitigate these challenges by increasing throughput and efficiency without compromising the integrity or decentralization of the underlying blockchain architecture. The core principle behind Layer 2 systems is to execute most transactions off-chain - that is, outside the main blockchain - and only settle final proofs or summaries back into the Layer 1 chain [15].

Several Layer 2 approaches have emerged, with Optimistic Rollups and zero-knowledge rollups (ZK-rollups) being among the most prominent. This section dis-

cusses four prominent Layer 2 approaches under these rollups and highlights the basic architectures.

2.5.1 Optimistic Rollups

Optimistic rollups are scaling solutions for faster and cheaper ways of making transactions on an Ethereum blockchain network, where Layer 2 blocks are optimistically assumed to be valid delegating block correctness on a fraud-proof mechanism [32]. It inherits the security and consensus of the Ethereum chain. Unlike the traditional Layer 1 blockchain, Optimistic rollups try to cut the transaction cost by executing as many calculations as possible off-chain reducing the computational load on the Ethereum mainnet and improving scalability. The primary concept is to bundle up transactions, compress data, off-chain computation, and then store the transitioned state on the Ethereum mainnet [39].The optimistic assumption makes the process more efficient and reduces the need for constant verification of every transaction. However, any suspicious transaction can be challenged during a fixed time window and can be deemed invalid by using fraud-proof mechanisms. If the rollup batch is not challenged during the challenge period, then it is considered valid and is accepted on Ethereum.

Sequencer: Also known as aggregator, collects all transactions and issues confirmations after executing all transactions off-chain. It publishes the new state root and transaction data in Layer 1.

Challenge Period: A fixed amount of time during which the sequencer or verifiers can challenge a transaction, and if they prove it invalid, the transaction will become invalid.

2.5.1.1 Optimism

Optimism (OP) is a scaling solution that aims to make the Ethereum network faster and cheaper. It was primarily designed to create a developer-oriented environment to create EVM compatible dApps and ultimately shrink the gap between Ethereum and Optimism. The long term vision of Optimism lies in creating a Super Chain of a collection of blockchains that are built on the OP stack, that will tackle the problems of congestion along with high transaction fees thereby increasing the overall throughput [44]. It uses fault proofs and also inherits the security of the Ethereum mainnet for safe and risk-free transactions and token management. Due to the blockchain trilemma, Optimism has to compromise the decentralization of the system to maintain scalability and security, as they have a centralized body, known as the sequencer that looks over the network. Currently, they are working on a functional fault proof system that will have a security council to supervise and intervene in case of disputes but the future goal is to have smart contracts fully managing the governance of the network without any centralized human intervention.

- **Fault Proofs:** Previously known as “fraud proof”, it refers to the process

that is followed when a transaction is challenged within the challenge window, i.e. currently 7 days [43].

- **State commitment:** The commitments that are published on Layer 1, that is Ethereum in the case of Optimism mainnet without any direct valid proof, which stays pending during the challenge period.
- **Verifier:** Handles rollup data and Layer 2 state transition while acting as proposers or challengers.

2.5.1.2 Arbitrum

Another Layer 2 scaling solution is Arbitrum [55]. It addresses blockchain's issues with high transaction fees and congestion. It has several advantages, including lower transaction fees compared to Ethereum mainnet, faster transaction processing and confirmation period, and the ability to maintain Ethereum's strong security. Moreover, it supports complex smart contracts and dApps, while ensuring high compatibility with Ethereum's existing architecture.

To simply state how Arbitrum works, it uses Optimistic Rollups to process transactions off chain. The transactions are bundled into batches and executed on Arbitrum's chain. Just the final results are posted to Ethereum, which saves space and cost. It initially assumes all transactions to be valid, but if someone suspects malicious behavior, the transaction can be challenged.

2.5.2 ZK-Rollups

ZK-rollup is a promising emerging category of Layer 2/off-chain approaches that could potentially be able to eliminate the final constraints to blockchain adoption in the Internet of Everything (IoE) [19]. ZK-rollups assume that all transaction computation data is false unless and until it is proven otherwise using validity or zero-knowledge proofs [29]. A way of demonstrating a statement's validity without disclosing the claim itself is called zero-knowledge proof [61]. The person attempting to prove a claim is known as the "prover," and the one confirming the claim is known as the "verifier." Through the use of a zero-knowledge protocol, one party (the prover) can demonstrate to another (the verifier) that a certain claim is true while withholding all other information. On top of that, ZK-rollups have less settlement time resulting in faster finality which improves the overall user experience [8]. Also, ZK-rollups eliminate trust by providing cryptographic certainty that the transactions are valid. All of these make ZK-rollups very compelling in a lot of areas while comparing with optimistic rollups. However, ZK-rollups are much more complex than optimistic rollups to work on. This makes it much harder for newcomers to work on this technology.

Validity Proof: Like optimistic rollups, ZK-rollups also use a proof called "validity proof". Which is a security paradigm for some Layer 2 solutions in which

transactions are bundled up into batches and sent to Ethereum as a single transaction in order to boost performance [61]. After the transaction computation is completed off-chain, a validation proof is sent to the main chain. By using this technique, more transactions are made feasible without compromising security.

2.5.2.1 ZKsync Era

ZKsync Era is a Layer 2 ZK-rollup, a trustless protocol that facilitates scalable and affordable Ethereum transactions through the use of cryptographic validity proofs [65]. In the ZKsync Era, most data is stored off-chain and computation is done off-chain. Before producing a validity proof, transactions are grouped into batches. The same security guarantees as in the Layer 1 are enjoyed by consumers because all validity proofs are validated on Ethereum. ZKsync was created while keeping in mind that it should look and feel like Ethereum with additional features such as lower transaction fees and higher throughput [65]. Any existing Ethereum private addresses work right out of the box with ZKsync. So we do not need to create separate accounts for this chain.

- **Zk-stack:** The framework ZKsync Era uses to build a secure protocol which is also scalable with higher throughput and faster finality.
- **zkEVM:** Heart of Zk-stack, which can execute transactions while being compatible with the ethereum ecosystem.
- **Elastic Chain:** The elastic chain is a network of ZK Chains (rollups, validiums, and volitions) that can be expanded forever, is mathematically secured, and functions flawlessly with a consistent, user-friendly interface [67].
- **Sequencer:** Nodes responsible for receiving transactions, executing them, combining them in a block and forwarding the block to a prover.
- **Prover:** Nodes responsible for generating cryptographic zero knowledge proof (ZKP) and sending them to the Layer 1 chain.
- **Blocks**: Consists of multiple transactions.
- **Batches:** Consists of multiple blocks.
- **Batches:** In blockchain systems, the term "finality" describes the time at which a transaction is regarded as irreversible and becomes a permanent part of the network [66].

2.5.2.2 Starknet

Starknet is a permissionless validity rollup that utilizes the STARK cryptographic proof. The system was built to achieve increased transaction throughput and low-cost transactions compared to Ethereum mainnet, while maintaining security. As

it uses ZK-rollups, it is more secure than other Layer 2 systems. A specialized programming language Cairo is used to build the Starknet OS and write scalable Starknet contracts. Starknet uses a virtual machine that utilizes Cairo language to generate validity proofs [17]. Starknet also has a native account abstraction to turn every account into a smart account and reduce reliance on private keys. The smart accounts are less reliant on protocol level and offer more customization opportunities to the developers. Scalable smart contracts and smart accounts makes Starknet one of the most developer friendly and promising platforms. One of the drawbacks of Starknet is its lack of decentralization at the present time. Currently it has only one sequencer node, controlled by a central authority [28]. However, the system is actively enhancing its decentralization along with improving throughput and transaction cost [25]. Thus, it is one of the most promising Layer 2 solutions.

- **Starknet gateways:** The gateways receive the transactions from the user and do a mempool validation check on these transactions. The transactions that pass the validation are marked as received. Then, it stores the passed transactions in a pool from which the sequencer can pull.
- **Mempool Validation:** A preliminary check on the account balance of the transaction sender, call data length, nonce, etc.
- **Sequencer:** An entity responsible for collecting validated transactions from mempool, ordering, batching, validating, executing transactions, and generating blocks. The sequencer validation is similar to mempool validation but more thorough.
- **Prover:** This entity receives blocks from sequencers and generates cryptographic proof. It sends the proof along with state change differences to Ethereum's main network.

Chapter 3

Literature Review

With the current Layer 2 solutions the entire blockchain system has made significant improvements to cope up with various existing issues. Each solution - whether it is a rollup, sidechain or any other mechanism, follows a different architecture that has been followed to achieve the most accuracy of each protocol. Every architecture follows a different strategy to optimize the whole system. Those architectural distinctions and overall discussion of each architecture we are working with, are discussed in section 3.1. And to further substantiate our research, we studied various publications of various authors that correlated to the systems we have discussed.

3.1 System Review

This section will look at each of the three protocols, Optimism, Arbitrum, and ZKsync Era, and explain their components and basic workflow.

3.1.1 Optimism

In Figure 3.1 we can get a primary understanding of the Optimism [44] network. Optimism aims to scale Ethereum without compromising its security by maintaining three essential properties, namely liveliness, validity, and availability. It guarantees liveliness by enabling a party to extend the rollup chain to accept a transaction within a fixed amount of time, therefore an actor is not allowed to block a transaction for more than a particular amount of time, which is acceptable by the community, making it practically useful. Validity is maintained as any party can initiate the execution of the rollup state transition function, which can also be validated by a smart contract on Ethereum. Finally, availability being a prerequisite for validity is referred to, any party being able to fetch the inputs required for the execution of the rollup state transition function correctly. Figure 3.2 shows the workflow of token deposit and transfer in the Optimism mainnet.

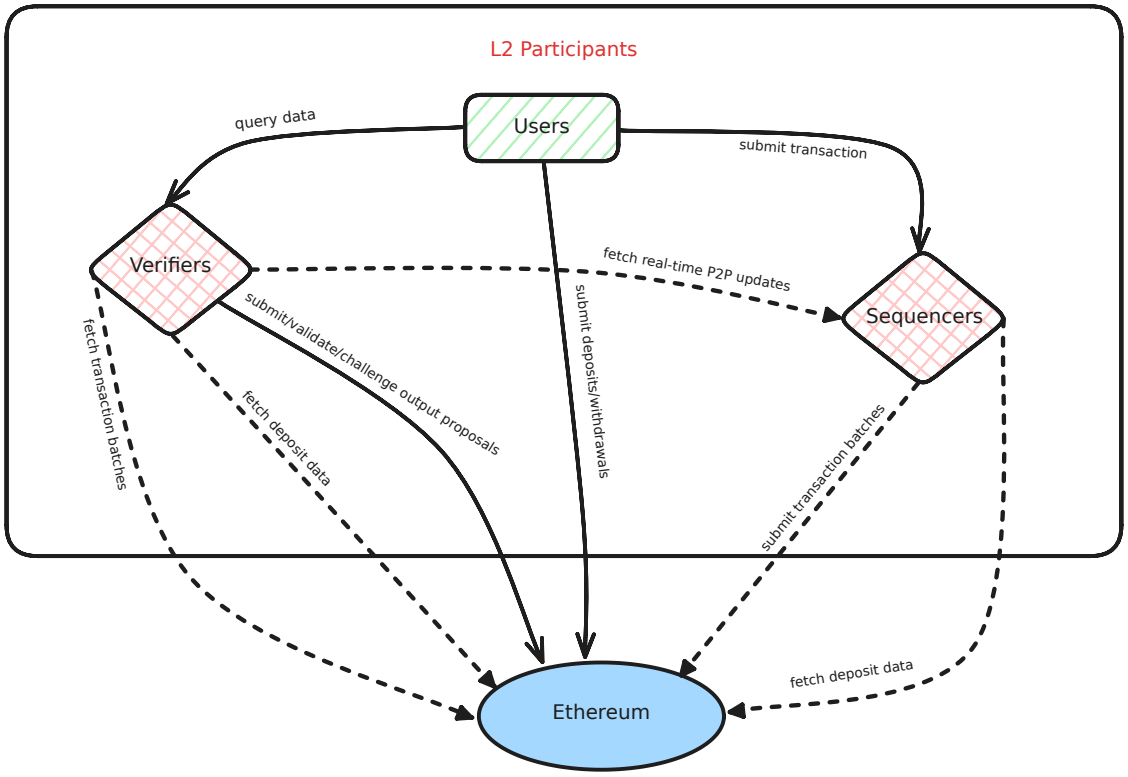

Figure 3.1: Optimism Network Participants Flow

The Optimism network primarily works due to the interaction of three types of actors or participants, namely users, sequencers, and verifiers who interact with the smart contracts on Ethereum and the Optimism mainnet to make the deposit, withdrawal, and transaction possible.

Users interact with contracts on Ethereum to submit transactions or they can also submit through a sequencer. They can also retrieve transaction data controlled by verifiers. The sequencer is the sole block producer in an Optimism chain; it might be a single sequencer or a consensus protocol of multiple sequencers, but it is regarded as a single entity in the network. Though the sequencer is not a trusted party, it plays a significant role for the Layer 2 chain to operate properly. The sequencer enhances the user experience by organizing transactions a lot faster and cheaper compared to users submitting all the transactions directly to Layer 1. Lastly, verifiers handle the Layer 2 estate transition, without the supervision or dependency of the sequencer, maintaining the integrity of the network by acting as proposers and/or challengers while serving the blockchain data to users. You will get a brief idea about the withdrawal process of Optimism by Figure 3.3.

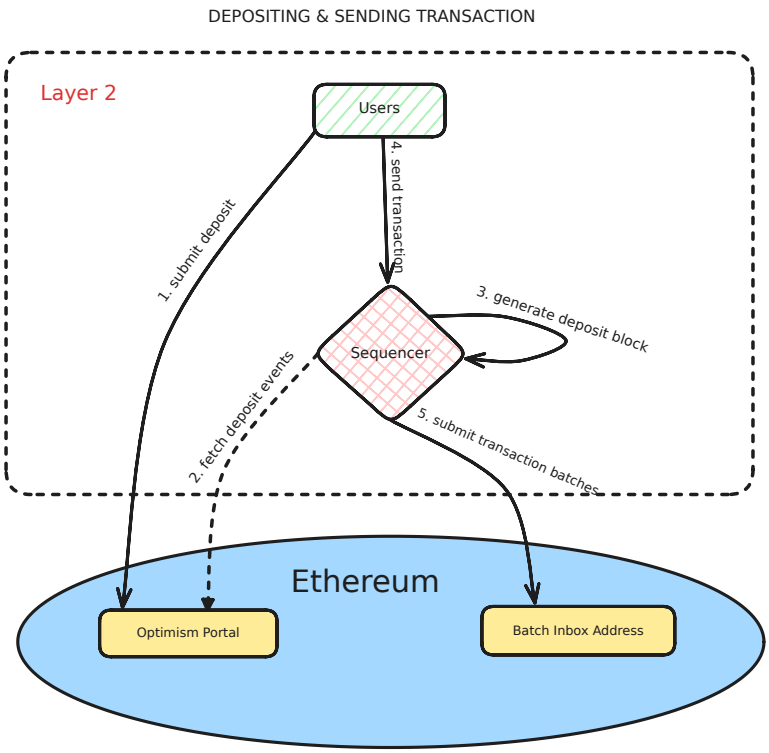

Figure 3.2: Optimism Deposit and Transfer Flow

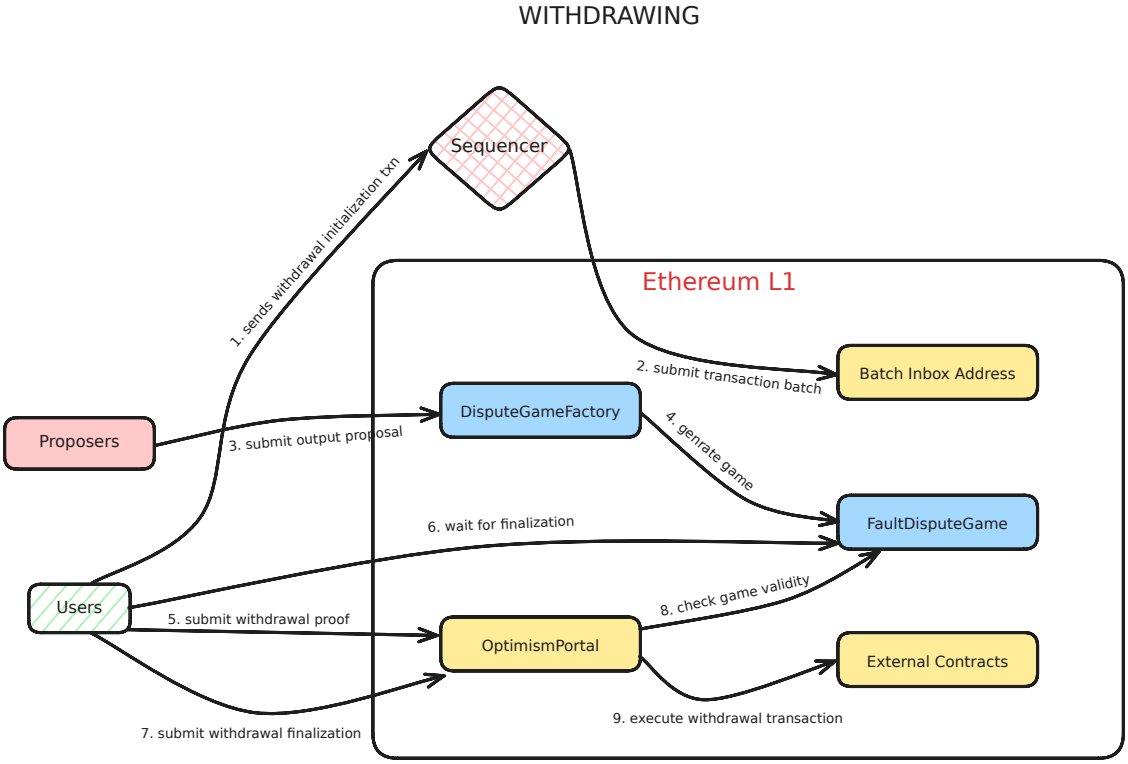

Figure 3.3: Optimism Withdraw Flow

3.1.2 Arbitrum

Arbitrum [55] is a technology suite made to scale Ethereum. Arbitrum chains can be used to do everything that can be done in Ethereum, but the transactions here will be faster and cheaper. Arbitrum uses Optimistic Rollups - a technique that processes transactions off the main Ethereum chain. It is designed to be fully compatible with EVM, and a recent version of Arbitrum, called Arbitrum Nitro, had its very own virtual machine, Arbitrum Virtual Machine (AVM). Batchers and validators are the two primary types of nodes which help run Arbitrum. They communicate with Ethereum's main chain, Layer 1, and make a different chain, which is Layer 2, which has its own state. Batchers take user's Layer 2 transactions and submit the data to Layer 1. What validators do is read Layer 1's transaction data and process the transaction, then the state of Layer 2 is updated. After that, the updated Layer 2

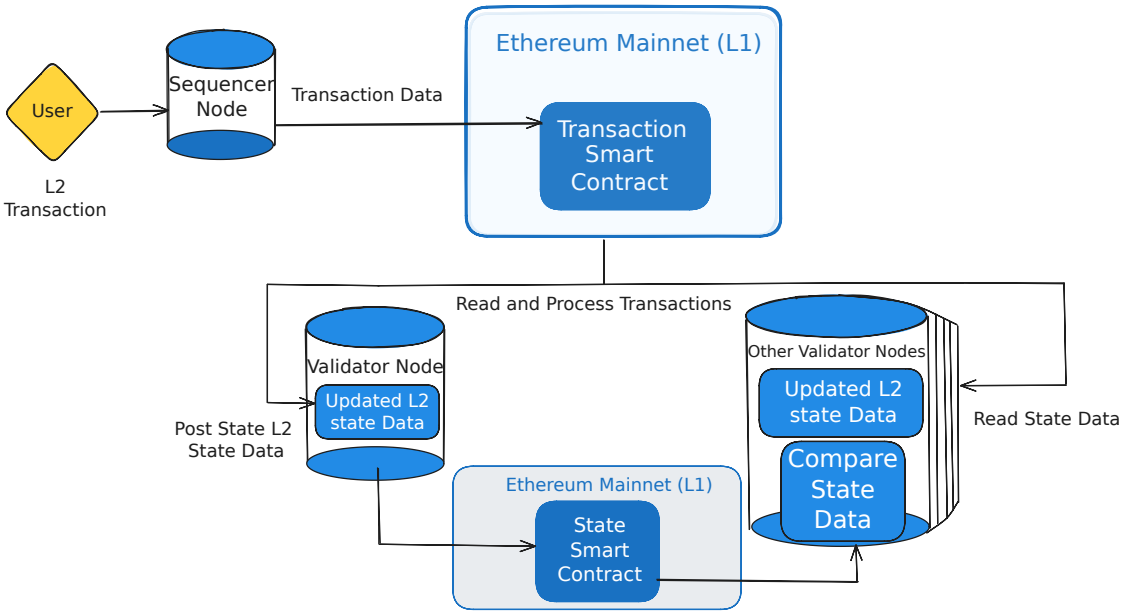

Figure 3.4: Arbitrum Workflow

state is posted to Layer 1, which means the new state can be verified by anyone. In Figure 3.4 we can take a look at the basic workflow. It starts when a batcher node, called the sequencer, receives Layer 2 transactions from the user. When enough transactions are received by the sequencer, they are bundled into a batch which is then posted to a smart contract in Layer 1. These transactions are read by a validator node and they are processed in the Layer 2 state's local copy. The new Layer 2 state root is compared with the original root from Layer 1 smart contract. In case any discrepancies are present, there will be a challenge in Layer 1. During the challenge, the validator and the challenger will, one by one, take turns and try to prove which state root is real. After the challenge, the loser's penalty is that their stake, or initial deposit, will be slashed. If the Layer 2 state root originally posted was not valid, future validators will destroy it and it will not be part of the Layer 2 chain.

3.1.3 ZKsync Era

ZKsync Era [63] developed a stack called ZK Stack for seamless future development in the ZK-rollup space. The stack is backed and secured by math instead of validators, increasing the reliability and trust while also maintaining the security of blockchain. The zero-knowledge Ethereum Virtual Machine (zkEVM), which is intended to carry out transactions while preserving complete Ethereum compatibility, is the central component of the ZK Stack [53]. ZK-rollups have the major benefit of being verifiable by outside validators [53]. ZK-rollups enable the external validation of its state through the proof supplied, in contrast to standard blockchains that require the operation of a full node for transaction verification. This helps maintain the integrity of the rollup easier and more effective. ZK-rollups can also be verified by other rollups, which makes it easier to build the elastic chain.

There are two major operators in the ZK Stack known as the sequencer and the prover [52]. The sequencer and the prover are responsible for creating blocks, proofs and then submitting them to the base layer or Ethereum by calling a smart contract in the Layer 1 chain. In order to update the Ethereum network's state, accounts send instructions known as transactions, which are cryptographically signed.

3.1.3.1 Transactions Workflow

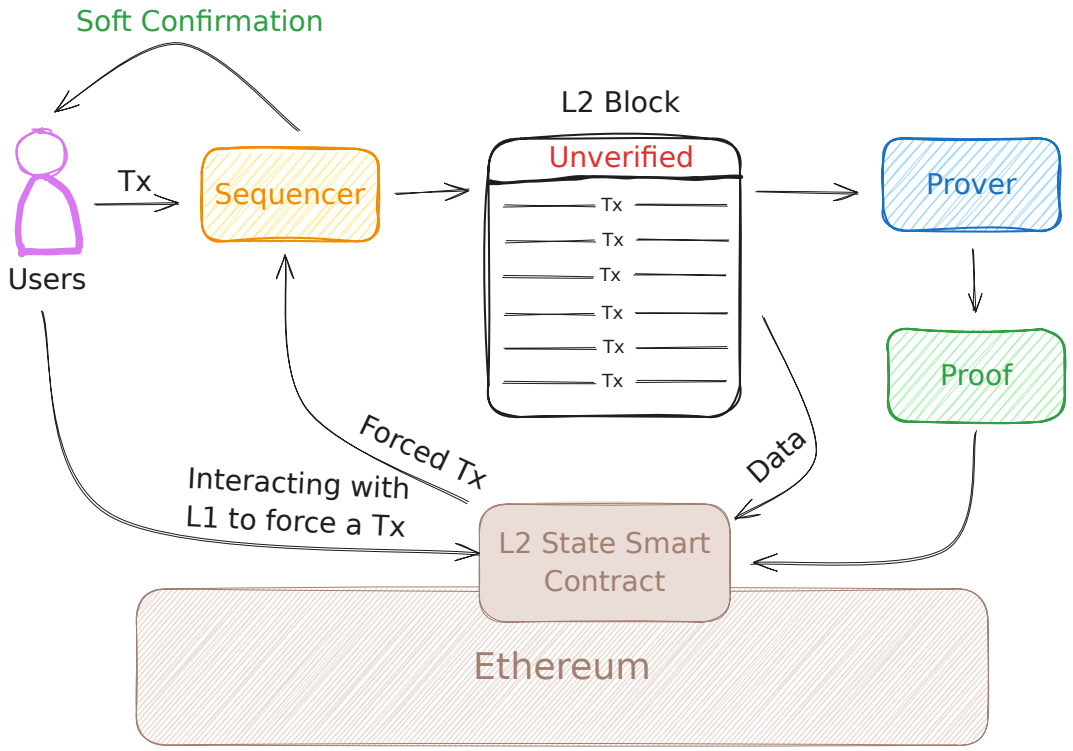

Figure 3.5: ZKsync Era Transaction Flow

The users can submit transactions to the sequencer or forcefully include it in the transactions by directly submitting them through Layer 1 [52]. Which makes the Layer 2 censorship resistant. The sequencer executes the transactions using zkEVM and returns a soft confirmation to the users. Then the sequencer bundles the transactions together and creates a block. After that, the sequencer sends optimized data to Layer 1 and forwards the blocks to the prover. The prover creates a cryptographics proof that the block has been executed. Then the prover sends the proof along with necessary data to Layer 1. Finally, the Layer 1 updates the rollup's state on the contract if the proof is valid. In Figure 3.5 we can see a visualization of how each component are working together.

3.1.3.2 Blocks and Batches

ZKsync Era uses blocks and batches in order to process the transactions in a more cost-effective and efficient way [51]. Layer 2 blocks, which are also called miniblocks, are unique to the ZKsync Era network and are not stored on the Ethereum blockchain. Processing of these blocks can be completed more quickly because they contain fewer transactions. In contrast, multiple successive Layer 2 blocks make up Layer 1 rollup blocks, or batches. All transactions from various Layer 2 blocks are compiled in these batches according to the order in which they were executed. The main goal of generating batches is to spread out the expenses of Ethereum interactions over a large number of transactions. In Figure 3.6 we can see a visualization of how different miniblocks create a batch.

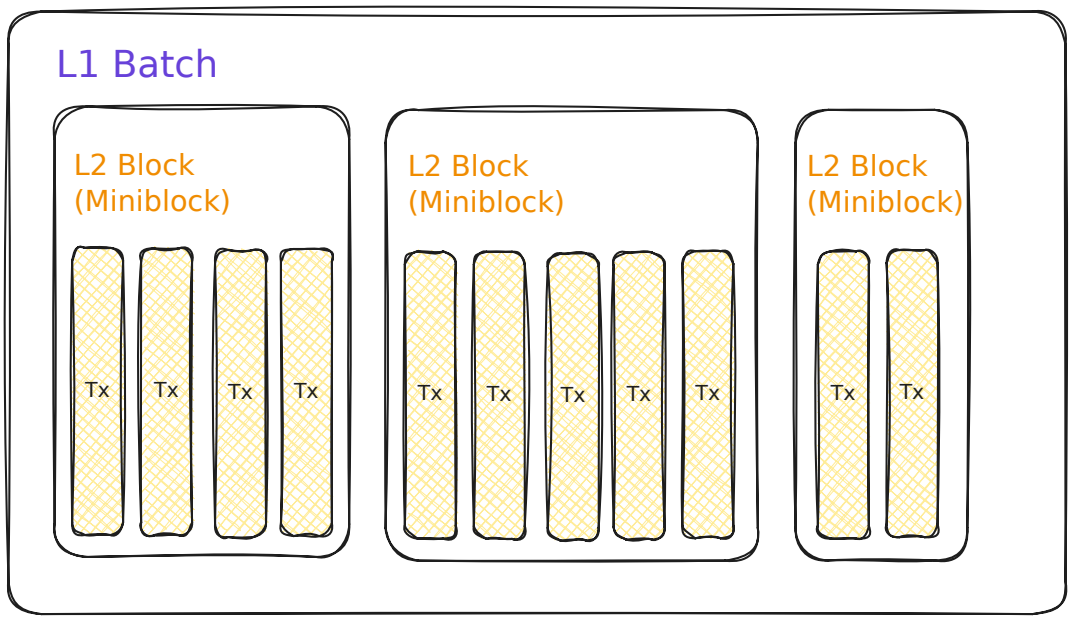

Figure 3.6: ZKsync Era Blocks And Batches

3.1.3.3 Finality

In Ethereum, when the network is working as expected, the finality time is generally two epochs, meaning around thirteen minutes [66]. ZKsync era inherits its security mechanism from Ethereum, as a result the finality time of ZKsync Era totally depends on Ethereum. It requires 5 steps for the ZKsync Era to reach finality of the transactions.

Transactions are first gathered and assembled into a batch [66]. Usually, this phase takes a few minutes. Second, the Ethereum network has the entire batch committed to it. Third, cryptographic evidence is produced to verify the batch as a whole. Usually, this procedure takes one hour. Fourth, an Ethereum smart contract receives the generated proof and verifies it. After one last inspection, the batch gets settled on Ethereum. This step involves a roughly 21-hour delay as a precautionary measure while ZKsync Era is in its alpha phase.

In Figure 3.7 we can see the finality process of the transactions. In general, a transaction on ZKsync Era has a finality time of approximately 24 hours, which corresponds to the finality of the associated Ethereum block [66]. However, transactions are considered instantly confirmed for better user convenience. Although this increases the overall user experience of the system, crucial transactions should be treated more cautiously, maybe waiting until the batch is confirmed.

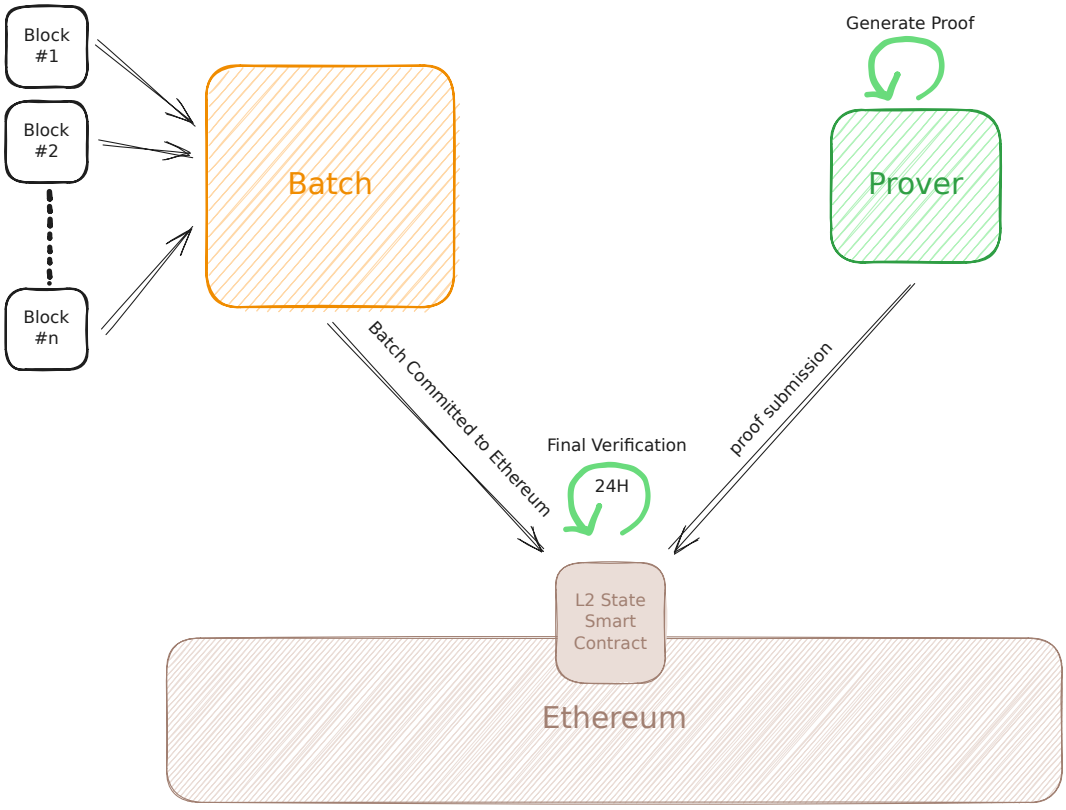

Figure 3.7: ZKsync Era Finality

3.1.4 Starknet

3.1.4.1 Transactions Workflow

At first, the user queries for the next available nonce, uses their private key to sign the transaction, and submit it to a starknet gateway to relay the transaction to Starknet network. Secondly, the gateway conducts a mempool validation. If it passes the validation it gets added to the mempool. Mempool is a temporary storage containing all transactions awaiting processing and mempool validation refers to checking account balance sender, calldata, nonce, etc. Third, sequencer nodes pull transactions from the mempool, run a validation check, execute the required work, and update the state. Sequencers are special nodes responsible for ordering and grouping transactions that occur within Starknet's Layer 2 environment. At present, there are a limited number of sequencer nodes, operated by a central entity. Fourth, following the state update sequencer adds the transaction into a block but emits the block at a later opportune moment. Emission of the block indicates acceptance on Starknet network. Finally, the prover node receives the block containing the transaction, re-executes the block, generates a proof, and sends it to Ethereum mainnet with state difference.

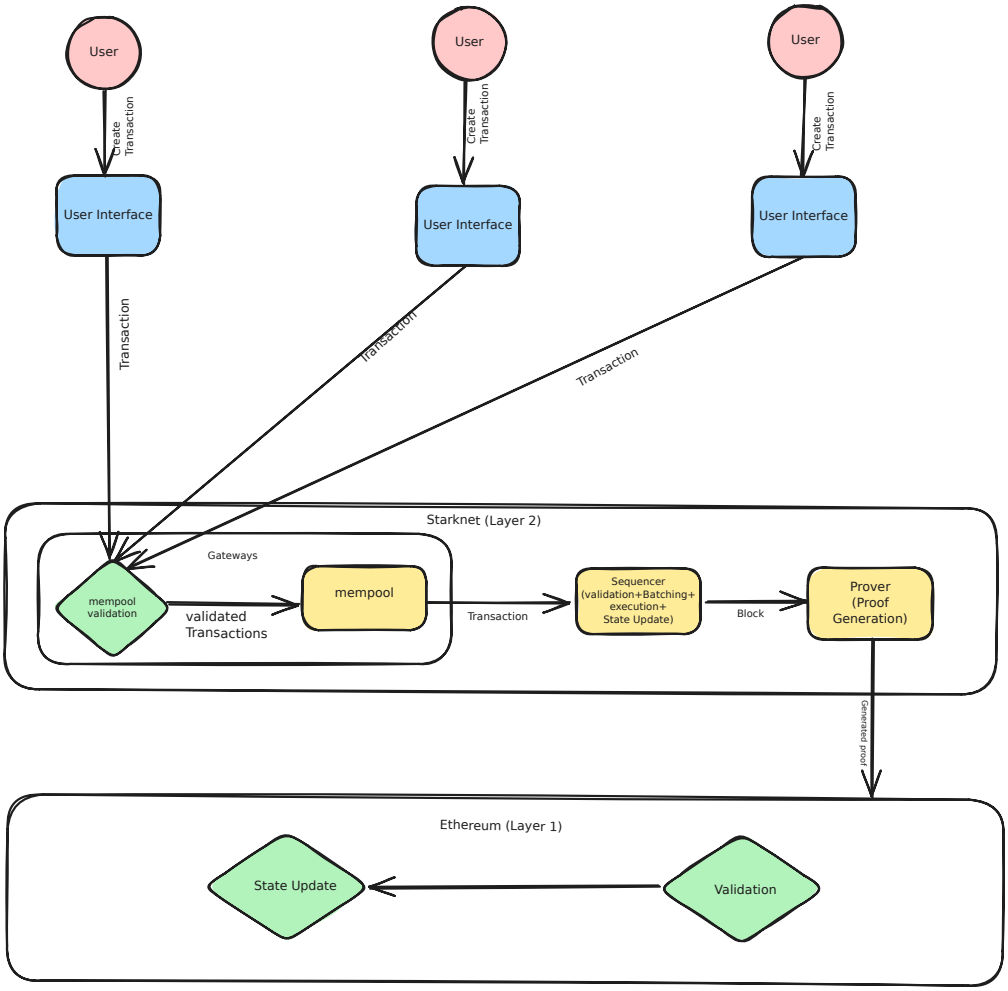

Figure 3.8: Starknet Trasaction Flow

In Figure 3.8 we can see a simplified transaction flow in starknet network and transaction submission in Ethereum mainnet.

Transaction Types: Three types of transactions can take place in Starknet:

(i) DECLARE: Create a new contract.

(ii) INVOKE: Trigger a function inside an existing contract.

(iii) DEPLOY_ACCOUNT: Deploy a contract to create a new account.

Gas Fees for Transactions: The fees are charged automatically during Layer 2 transaction execution. The Starknet OS sends ERC-20 tokens, with the transaction submitter as the sender and the sequencer as the receiver. The fee calculation varies based on the type of data being transmitted.

3.1.4.2 Blocks and Batches

Block Structure: Each block on the starknet contains parent_block_hash, block_number, global_state_root, sequencer_address, block_timestamp, transaction_count, transaction_commitment, event_count, event_commitment, l1_gas_price, l1_data_gas_price, l1_da_mode, protocol_version.

global_state_root: It represents the state commitment or a digest that represents the state after the commitment of the block.

transaction_commitment: A cryptographic commitment of the transactions included in the block, can be used by verifiers to match against actual transactions to check authenticity.

Event: Events refer to a mechanism for emitting and logging important occurrences or notifications that happen during the execution of smart contracts or applications on the Starknet.

l1_gas_price: Cost of storage updates and Layer 1 to Layer 2 messaging, set by Starknet protocol.

l1_data_gas_price: The storage update cost if l1_da_mode is set to BLOB. BLOB or Binary Large Object is a data type that can be attached to blocks but do not remain attached to those blocks permanently unlike traditional transactions.

3.1.4.3 Finality

The finality status for transactions in Starknet shown in Figure 3.9 are following:

Not Received: The transaction is not known by the sequencer node.

Received: The transaction has been Received by the mempool.

Rejected: The transaction failed sequencer validation, and did not get included in a block.

Accepted_on_l2: The transaction was successfully executed and entered into a Layer 2 block

Accepted_on_l1: The transaction was accepted on Ethereum mainnet.

There are two execution statuses for the transactions.

Reverted: The transaction passed sequencer validation but failed execution, and it gets included in block with status reverted. The senders account nonce increases, fees are charged till the point of failure, and state is partially reverted.

Succeeded: The transaction has been successfully executed by the sequencer and has been included in a Layer 2 block.

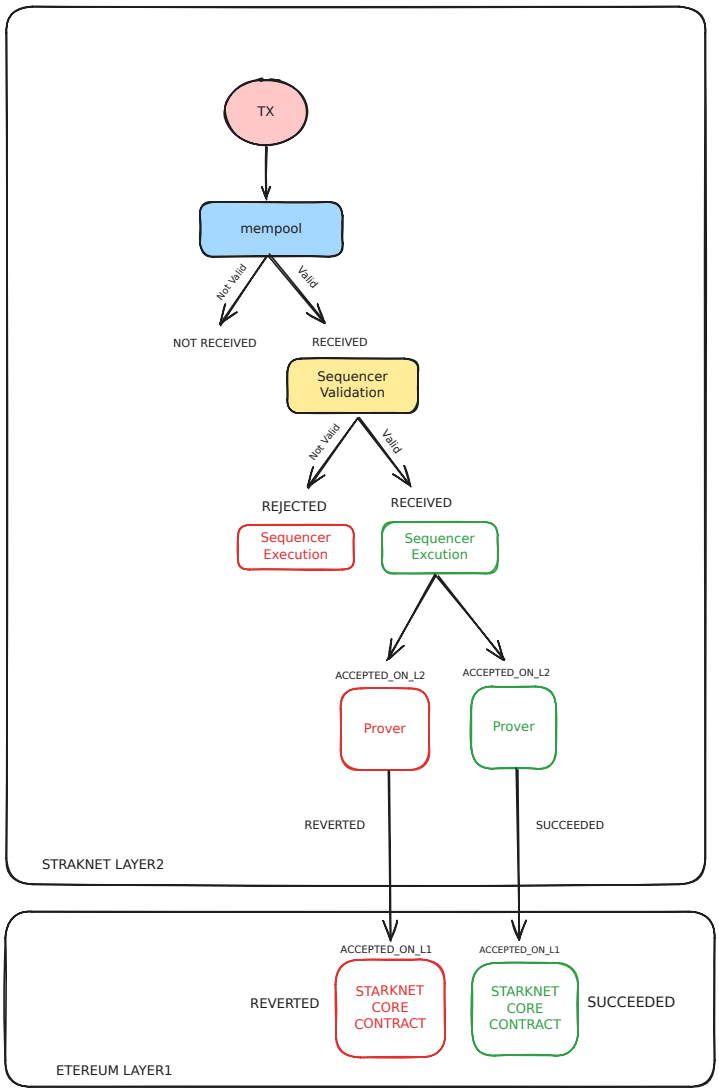

Figure 3.9: Starknet Transaction Status

3.2 Academic Review

Blockchain technology emerged as a decentralized alternative to traditional financial systems but was soon confronted with a critical challenge: scalability. Song et al. [48] highlight that Layer 1 blockchains such as Bitcoin and Ethereum, by requiring each node to validate and store all transactions, ensure strong decentralization and security - but this design severely limits throughput. Ethereum, even with its transformative Ethereum Virtual Machine (EVM), processes only about 12-15 transactions per second (TPS), which is insufficient for high-volume, real-time applications [18], [48]. Gangwal [18] and Yoo [30] argue that while initial solutions such as increasing block size, transitioning to proof-of-stake, or sharding offer marginal improvements, they cannot fundamentally resolve the scaling dilemma inherent in Layer 1.

This realization redirected focus to Layer 2 solutions, which perform transactions off-chain while leveraging the main chain for final settlement. Thibault et al. [21] explain that rollups have become a central pillar of Ethereum's scalability strategy. Optimistic rollups, according to Gangwal [18], assume transaction validity unless proven otherwise and offer lower cost and ease of implementation, though they suffer from long withdrawal delays due to fraud challenge windows. ZK-rollups, as advocated by Lavaur et al. [19], generate succinct zero-knowledge proofs to validate transaction batches, offering faster finality and improved privacy. Thibault et al. [21] add that these ZK-based solutions, while computationally demanding, present promising scalability benefits and stronger security guarantees.

Yoo [30] classifies scaling approaches into centralized validator systems (like EOS.IO and Algorand), Layer 1 structural upgrades (such as Ethereum 2.0 and Bitcoin Cash), and Layer 2 protocols (like Arbitrum, Optimism, and ZKSync). Rebello [45] extends this framework, offering a four-tiered model that examines scalability through hardware, networking, Layer 1, and Layer 2. He stresses that improvements must be coordinated across these layers for meaningful gains. According to Rebello, while Layer 2 solutions provide impressive throughput enhancements, their long-term impact depends on addressing interoperability and network congestion issues holistically. On the other hand, sidechains present another model, with Neiheiser et al. [26] emphasizing their flexibility in consensus design and speed. Protocols such as Polygon, Ronin, and Gnosis utilize sidechains to reduce congestion on the mainnet. However, Neiheiser cautions that sidechains trade off some decentralization and security due to their reliance on independent validators. Channels, like those used in Lightning Network and Raiden, allow two parties to transact off-chain efficiently, but as Neiheiser notes, these are more niche solutions suited to repetitive or micropayment scenarios. Despite their limitations, these channels are often praised for their low fees and latency in specific use cases. Authors like Thibault et al. [21] further evaluate ZKsync and Starknet, both advancing general-purpose zero-knowledge rollups. They note issues such as EVM incompatibility and high resource requirements for proof generation as barriers to widespread use. Despite this, they argue that the long-term scalability and privacy benefits justify continued investment in these technologies. Lavaur et al. [19] also emphasize the applicability of ZK-rollups in the Internet of Everything (IoE), where data volume and

latency requirements make Layer 1 infeasible. In these environments, the ability to prove validity off-chain and anchor data on-chain securely is not just beneficial but essential.

Privacy, as Ishii [57] discusses, is not a peripheral issue but one deeply tied to scalability and user adoption. While public blockchains provide transparency, many real-world applications require confidentiality. Ishii explores solutions such as zk-SNARKs and zk-STARKs, alongside higher-level tools like zkay and ZeeStar, which aim to simplify private smart contract development. He also discusses Aztec 2.0 as a Layer 2 privacy-focused protocol but warns of trade-offs, especially when using hardware solutions like Intel SGX that introduce new trust vectors. Ishii emphasizes that Layer 2 can enable privacy without compromising decentralization, but only with careful system design and cryptographic validation.

Addressing data availability is another core issue. Saif [47] contends that rollups rely heavily on timely and reliable access to off-chain data. Without robust data availability mechanisms, the security assumptions of rollups falter. His comparative study of Starknet, Optimism, and ZKsync highlights how each protocol navigates this trade-off between scalability and data integrity. Saif emphasizes that data availability layers must be tightly integrated with rollup protocols to avoid bottlenecks that would otherwise negate their performance gains. Makhdomi and Gillani [58] provide a practical case through their rollup-based decentralized ridesharing system, which outperforms traditional Ethereum implementations but also faces integration and complexity challenges. Their work underscores the real-world applicability of Layer 2 scalability methods and the potential for user-facing applications to benefit from significant improvements in transaction cost and confirmation time. Moreover, they argue that as decentralized apps continue to grow in user base and feature complexity, the demands on blockchain infrastructure will only intensify, further necessitating Layer 2 advancements.

In a forward-looking perspective, Thibault et al. [21] propose a modular blockchain framework that separates execution, consensus, and data layers. This architecture allows Layer 2 systems to act as native components of the blockchain ecosystem rather than mere extensions. It also supports flexibility and performance optimization across diverse use cases. Yoo [30] echoes this sentiment, stating that the future of blockchain scalability depends on composability between layers rather than attempting to over-engineer Layer 1. They also emphasize that incentive alignment across these layers is critical, particularly for security, as poor incentives could undermine the reliability of even the most technically sound protocols. As Yoo [30] and Gangwal [18] point out, no single solution can fully resolve the blockchain trilemma. However, continued progress in Layer 2 protocols—whether through rollups, sidechains, channels, or privacy-enhancing cryptographic designs—brings us closer to a blockchain infrastructure capable of supporting secure, scalable, and decentralized global applications. The path ahead may be complex, but with persistent innovation and thoughtful engineering, the long-standing challenges of blockchain scalability are steadily being addressed. As the community advances from theory to implementation, and as user expectations grow, the maturity of Layer 2 protocols will likely determine the pace of blockchain's integration into mainstream digital infrastructure.

Chapter 4

Proposed System

The core objective of our research is to design and develop a Layer 2 Evaluation Framework focused at helping businesses and developers choose the most appropriate Layer 2 scaling solution for their specific use cases, businesses, or protocols. The system will analyze and compare performance, security, interoperability, cost, and developer experience metrics across multiple Layer 2 solutions. Initially, we have explored five different solutions: Arbitrum, Optimism, Polygon PoS, ZKsync Era, and Starknet. For the later part of our research we have narrowed down our focus to Arbitrum, Optimism, ZKsync Era and Starknet to conduct a more comprehensive research. We have chosen not to include Polygon PoS in our evaluation of Layer 2 scaling solutions, as it fundamentally differs from rollup-based architectures in both design and trust assumptions. Unlike optimistic rollups or ZK-rollups, which inherit security from Ethereum's base layer by relying on Ethereum for data availability and finality, Polygon PoS operates as a sidechain with its own validator set and consensus mechanism. This introduces a separate trust model and creates a potential single point of failure, as the security of the network depends on the honesty and liveness of Polygon's validators rather than Ethereum's consensus layer [10].

4.1 Methodology

As part of this research, we first conducted a literature review of relevant academic articles related to Layer 2 technology and protocols. As the field is relatively new, the opportunity to filter work was minimal for us. Following that, we conducted a system review of prominent Layer 2 protocols of Ethereum. We selected the protocols based on their TVS. After that, we worked on proposing a system capable of ranking Layer 2 protocols based on user preferences. In the system proposal, we selected some core criteria based on the fundamental characteristics of blockchain and mapped relevant heuristics to those core criteria. Based on these we created the main evaluation formula and extracted values of all heuristics for our selected four Layer 2 protocols from L2BEAT. Following that, we generated a hypothetical scenario of a user giving preference and calculated evaluation score for all four protocols based on the proposed evaluation formula. Finally, based on this the ar-

chitecture and protocol flow of the evaluation application was created. Based on the architecture and protocol flow the application was developed and tested.

4.2 System Proposal

Although we have only chosen some of the most popular L2s (in terms of TVS) at the time of the study (2023-2024), our goal is to keep our study methods as modular as possible and leave the community with a process that is easy to replicate for any Layer 2 protocols. Our designed framework will work as a must have tool to make a decision while deciding which Layer 2 solution or solutions might be most suitable for a particular use case. The framework will provide a data-driven analysis of some of the most prominent Layer 2 protocols consisting of optimistic rollups and ZK-rollups. The framework will consider multiple factors such as performance optimization, cost reduction, high security, etc. while evaluating a protocol and try to optimize all the crucial factors as much as possible. However, the framework must not significantly compromise any core components while trying to optimize others.

We will also provide a user-friendly interface to make the framework more accessible, where users can input their requirements (e.g., transaction volume, latency tolerance, security needs) and receive a tailored recommendation based on real-world data from various block explorers, protocol documentation, and live tests.

This system will not only focus on the technical performance of the protocols but also take into account factors like developer tooling, ease of integration, and long-term sustainability. Once developed, the framework will be tested in a real-world scenario using a prototype application, allowing us to validate the claims and performance metrics of each protocol under typical operational conditions.

After introducing the system, it becomes essential to analyze the requirements needed to develop such a comprehensive framework.

4.3 Requirement Analysis

4.3.1 Functional Requirements

I Users are provided with a form where they rate various parameters which represents the users' priorities according to the requirements of the system that they are developing.

II The system then assigns weights to the respective evaluation according to the user preference that was taken from the form.

III The system then performs the evaluation of the four protocols and ranks them accordingly.

IV The users connect their metamask wallet to the system.

V The users can then deploy their smart contract to any of the following Layer 2 blockchains using a deploy button beside the protocol of their choice.

VI Finally, the users are provided with a confirmation message and the information related to the deployment.

4.3.2 Non Functional Requirements

I The system should provide an intuitive interface for ease of use.

II The system should be easy to use for users with minimal knowledge about Layer 2 protocols.

III The generated score of each protocol should be presented in a simplistic manner for easy comparison.

IV The system should be able to take user's sensitive information like private key in a manner so that it is not disclosed to anyone.

4.4 Proposed Framework

Our system will work based on the analysis of five core criteria, which are scalability, security, decentralization, cost efficiency, and developer experience. The user will provide their preference on a specific criteria, as elaborated in Section 4.4.1.1 on a scale of one to ten. The user's input based on their particular use cases will work as weight for each criteria, and will be used to determine the final score of each of the protocols. Each criteria has relevant features mapped to it. Most features are taken from L2BEAT, a reputable website that provides information about layer two protocols to ensure dynamic data upgradability of the system. Each of the weights will be multiplied by the sum of each of the normalized values of features mapped to that particular criteria. All of these will come together in the following equation to find the final score of each Layer 2 protocol:

$$\text{FINAL SCORE, } \mathrm{P}_k = \frac{\sum_{i=1}^{n} W_{i,k} \cdot F_{i,k}}{5}$$

Where,

$$F_{i,k} = \frac{\sum (\text{All Mapped Heuristics})}{\text{Total Number of Heuristics}}$$

Here, W is the weight retrieved from user input to a core criteria, F is the average value of all heuristic mapped to that core criteria, i is indicating each core criteria, and k is indicating the protocols.

4.4.1 User Preference Analysis

Understanding the user preference is one of the key jobs that is needed to be accomplished for designing a system. To ensure that our evaluation system aligns with the user's expectations and provides service according to his/her needs, we have selected a few specific core criteria that will help us understand what the user needs as a service provider. After going through the documentation and guidelines present on websites, the whitepaper, yellow paper of Ethereum, and other Layer 2 solution protocols, the following are the key parameters that can help us understand how our users want to prioritize various factors according to their needs:

4.4.1.1 Scalability

Scalability is one of the properties mentioned in the blockchain trilemma. It refers to the ability of a blockchain system to handle a large number of transactions within a fixed timeframe [2]. It is essential for blockchain systems to be scalable in order to meet real life applications. That is why scalability is a core criteria in this framework, mapped to the heuristic UOPS and liveness described in the heuristic section.

Evaluation Scale: Using a scale from 1 (minimal) to 10 (high) provides a flexible method for assessing the scalability requirements of a dApp.

4.4.1.2 Security

Security is also one of the properties mentioned in the blockchain trilemma [2]. It is defined as the ability of the blockchain system to be secured from attackers with a certain amount of computational power. The security of a blockchain system is not only responsible for stopping attackers from corrupting transactions but also blocking chain state manipulation. Thus, security has been taken as a core criterion for our evaluation framework and users are able to assign a value of 1 to 10 as per the scale of security required by their system. This core criterion is mapped to the heuristic risk analysis, TVL, and finality time described in the heuristic section.

Evaluation Scale: A standardized numeric range from 1 (minimal) to 10 (high) provides an objective means of assessing the platform's security robustness.

4.4.1.3 Decentralization

Decentralization refers to a blockchain system's ability to operate at full capacity in a scenario where each participant of the system has a capability less than the full capacity of a system [2]. It is also a property of the blockchain trilemma and essential at ensuring that no single entity can manipulate the system for its own benefit. Thus, decentralization is being considered as a core criteria in our evaluation framework. This core criterion is mapped to the heuristic rollup stage and operator

mode described in the heuristic section.

Evaluation Scale: A formalized scoring system from 1 (minimal) to 10 (high) is applied to assess the decentralization architecture, including validator distribution and governance transparency.

4.4.1.4 Cost Efficiency

Cost is a significant factor in public blockchains. As the system is built around a reward mechanism, it costs to process transactions as well as to maintain the whole system. These costs are mainly a burden to most users despite being a necessary part of the system. One of the main motivations of Layer 2 protocols was to reduce usage cost [56]. That is why cost efficiency is selected as a primary criterion for the evaluation system. This core criterion is mapped to the heuristics on chain cost described in the heuristic section.

Evaluation Scale: Cost efficiency is assessed on a numerical scale from 1 (minimal) to 10 (high), enabling comparative analysis across different blockchain protocols.

4.4.1.5 Developer Experience

The blockchain technology was first introduced by Satoshi Nakamoto the inventor of bitcoin in 2008 [9]. Thus the technology itself is quite new and Layer 2 protocols were introduced much later in an attempt to solve the limitations of the blockchain technology. Thus, the amount of skilled personnel in this field is very limited [23]. Moreover as the Layer 2 protocols are very new, it has much less skilled personnel. That is why developing experience of a Layer 2 protocol has been selected as a core criterion in the evaluation system. This core criterion is mapped to EVM compatibility and programming language described in the heuristic section. Evaluation Scale: A structured scale from 1 (minimal) to 10 (high) facilitates a standardized assessment of how conducive a platform is to developer productivity and integration efficiency.

4.4.2 Heuristic Analysis

Heuristic analysis is an important step in the design of our system. Unlike user preference analysis, which is based on subjective opinions, heuristic analysis focuses on the objective functionality and efficiency of the system. These metrics are derived from data collected through reliable analytics on Ethereum Layer 2 protocols. For the first set of parameters - Total Value Stored (TVS), Risk, Liveness and User Operations Per Second (UOPS) - the data were collected from L2BEAT within the time frame of 180 days from March 13, 2025 to June 11, 2025. we normalized the

values using the following formula:

$$\text{Normalized Value} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{4.1}$$

Here, x is the value of the parameter being normalized, while $x_{\min}$ and $x_{\max}$ are the minimum and maximum values for that parameter across all protocols, respectively. Note that the best possible value is also considered during this process.

For the remaining parameters - Onchain Cost, Finality Time, Avg. Transaction Submission Interval and Avg. State Update Interval - lower values are preferred. Therefore, we subtract the normalization value from 1 as shown in the formula:

$$\text{Normalized Value} = 1 - \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{4.2}$$

This ensures consistency across all normalized metrics by aligning them such that higher values indicate better performance. We applied this normalization technique to each of the above parameters for all considered Layer 2 solutions.

4.4.2.1 Risk Analysis

In order to find the security-related risks associated with Arbitrum, Optimism, Starknet, and ZKsync Era, we conducted a security threat analysis. To perform threat analysis, we followed the risk analysis conducted by L2BEATs for each protocol. It includes five aspects, namely state validation, data availability, exit window, sequencer failure, and proposer failure. State validation determines how the network ensures that Layer 2 state transitions are legitimate. For which Optimism and Arbitrum use fraud proofs and ZKsync Era uses zero-knowledge proofs for validation. Data availability refers to whether users can independently access all the data required to verify Layer 2 states - each of the three protocols posts transaction data on Ethereum Layer 1, ensuring verifiability, though implementations may differ. Exit window is the timeframe available to users to safely withdraw their assets if a protocol change is disputed or under attack; Arbitrum provides a 10 day window, Optimism and ZKsync Era does not support any exit window which could turn out to be an imminent threat for the transaction. Sequencer failure, which could prevent transaction inclusion. As Arbitrum allows users to force transactions through Layer 1 with some delay, Optimism and ZKsync Era let users submit to a Layer 1 queue but lack mechanisms to enforce processing, giving sequencers significant control. Lastly, Proposer failure examines how the system handles situations where the designated actors responsible for submitting state roots become inactive; Arbitrum and Optimism have fallback mechanisms that allow anyone or users themselves to propose roots without any permission, whereas ZKsync relies on a whitelist of proposers - if they fail, withdrawals are frozen, indicating centralization concerns. Together, these five aspects provide comprehensive data regarding the risk associated with each protocol for evaluating the systemic resilience, highlighting the delicate balance between scalability, decentralization, and user trust that each design must navigate.

4.4.2.2 TVS

Total Value Secured or TVS measures bridged and natively minted assets, which are not necessarily locked and the amount of funds locked in a protocol's canonical bridges. It serves as a proxy for the protocol's user trust, adoption, and liquidity. Understanding TVS helps users, investors, and analysts assess a protocol's adoption, security, and reliability. A higher TVS suggests that users are confident in the security and utility of the platform, and it also ensures deeper liquidity pools for smoother transactions with minimal slippage. The TVS data was collected from L2BEAT [40]. Since TVS is calculated in USD, in order to normalize it we took the maximum and minimum TVS from our set of protocols and applied a min-max normalization formula.

$$\text{Normalized TVS}_i = \frac{TVS_i - TVS_{\min}}{TVS_{\max} - TVS_{\min}}$$

4.4.2.3 UOPS

UOPS refers to user operations per second. In recent days it is being used in the place of TPS or Transactions Per Second, which was a fundamental metric for evaluating the scalability and transaction-handling capacity of a Layer 2 solution. UOPS utilizes ERC-4337 which is Ethereum's standard account abstraction. It is basically a pseudo transaction that gives smart contract information about users' desired actions [50]. UOPS reflects how many of these user intents a network can process per second, providing a more application-level view of scalability. A high UOPS value indicates that the network can efficiently handle a large volume of user actions, making it well-suited for high-traffic DApps. In this case the maximum and minimum value were taken based on the highest and lowest UOPS found in this time period out of these 4 protocols. (Arbitrum had the highest and ZKSync had the lowest)

$$\text{UOPS}_i = \frac{\text{UOPS}_{i,\max} + \text{UOPS}_{i,\min}}{2}$$

$$\text{Normalized UOPS}_i = \frac{\text{UOPS}_i - \text{UOPS}_{\min}}{\text{UOPS}_{\max} - \text{UOPS}_{\min}}$$

4.4.2.4 Finality Time

Finality time refers to the time period required by a blockchain protocol to make a transaction irreversible and permanent. The finality time used by the evaluation system was normalized by the min-max normalization and later modified by subtracting from 1 as lesser finality time is more preferable.

$$\text{Normalized Finality Time}_i = 1 - \frac{\text{Finality Time}_i - \text{Finality Time}_{\text{min}}}{\text{Finality Time}_{\text{max}} - \text{Finality Time }_{\text{min}}}$$

4.4.2.5 Onchain Cost

The onchain cost refers to the cost Layer 2 protocols pay to Layer 1. The evaluation system utilizes average transaction cost per Layer 2 user operation and these values were taken from L2BEAT with the timeframe of 180 days (2025 March 13 to 2025 June 11). The values were provided in USD, in order to normalize it we took the maximum and minimum average transaction cost per Layer 2 user operation from our set of protocols and applied a min-max normalization formula.

$$\text{Normalized Onchain-cost}_i = 1 - \frac{\text{Onchain-cost}_i - \text{Onchain-cost}_{\text{min}}}{\text{Onchain-cost}_{\text{max}} - \text{Onchain-cost}_{\text{min}}}$$

4.4.2.6 Liveness

Liveness is an indicator of the activity frequency of the protocol operators. It refers to how frequently the operators of the protocols submit the transactions. The values of this heuristic were also collected from L2BEAT. There are two values of this heuristic which are average transaction submission interval and average state update interval. To use these values in the evaluation protocols both were normalized using the min-max normalization process and then an average value was generated.

$$\text{Normalized ATSI}_i = 1 - \frac{\text{ATSI}_i - \text{ATSI}_{\text{min}}}{\text{ATSI}_{\text{max}} - \text{ATSI}_{\text{min}}}$$

$$\text{Normalized ASUI}_i = 1 - \frac{\text{ASUI}_i - \text{ASUI}_{\text{min}}}{\text{ASUI}_{\text{max}} - \text{ASUI}_{\text{min}}}$$

$$\text{Liveness} = \frac{\text{Normalized ATSI}_i + \text{Normalized ASUI}_i}{2}$$

Here, Average Transaction Submission Interval (ATSI) refers to the average time taken for a transaction to be submitted to the Layer 2 protocol, after it is initiated by the user and Average State Update Interval (ASUI) refers to the average time taken to finalize a Layer 2 batch of transactions and update the changes in Layer 1.

4.4.2.7 EVM compatibility

EVM or Ethereum Virtual Machine compatibility of a system makes it easy to adapt for Ethereum users and developers, as these share the same peripheral tools and

applications as Ethereum [22]. For the evaluation framework, an EVM compatible Layer 2 protocol gets a score of 1 while a non-EVM compatible protocol gets a 0 on this heuristic.

4.4.2.8 Programming Language

As Ethereum smart contracts and dApps are majorly coded in solidity, it can be assumed to be a standard language for Ethereum related protocols. Thus, our evaluation system analyzes the programming language of Layer 2 protocols and assigns a score of either 1 if the Layer 2 protocol uses solidity to write programs or 0 if it uses any other language.

4.4.2.9 Stages

The stages of L2BEAT [40] categorize Layer 2 rollups based on their level of decentralization, security guarantees, and reliance on Ethereum for user safety.

Stage 0 rollups are early or partially implemented systems that still depend heavily on centralized actors for key functions such as censorship resistance, while posting data and state roots to Ethereum. These systems often fall short of L2BEAT's principle that only a compromise of 75% or more of the security council (excluding protocol bugs) should be able to permanently prevent a message from moving from Layer 2 to Layer 1 or allow an invalid one to be accepted (L2BEAT - stage 0).

Stage 1 rollups take a step forward by ensuring users can exit the system without the permissioned operators. Fraud or validity proofs are submitted by multiple external actors, and users are guaranteed a minimum exit window (typically 7 days) if the system changes. However, some operations, such as upgrades unrelated to bugs, can still be executed by centralized actors with less than 30 days' notice to users, limiting full trust minimization.

Stage 2 rollups are designed to closely match Ethereum's security standards. In these systems, the only way to permanently block or alter a message from Layer 2 to Layer 1 is by compromising at least 75% of the security council. All critical functions - such as proof generation, node operation, and upgrades must be decentralized, and users must always have sufficient time (e.g., 30 days) to exit in case of adverse changes.

In our system, we decided to assign scores to each stage as follows:

- Stage 0 $\rightarrow$ 0.0
- Stage 1 $\rightarrow$ 0.5
- Stage 2 $\rightarrow$ 1.0

Decentralization is calculated solely based on these stage values.

4.4.2.10 Comparison Across Key Metrics

In Table 4.1, we can see the comparison between our selected protocols across the metrics we are using to evaluate the final score.

Feature	Optimism	Arbitrum	Starknet	ZKsync Era
TVS	3.06B	13.65B	511.52M	468.63M
TVS Normalized	0.197	1	0.00325	0
UOPS	15.575	26.765	9.915	1.45
UOPS Normalized	0.558	1	0.334	0
Onchain Cost	0.002586	0.004009	0.000598	0.006652
Onchain Cost Normalized	0.328	0.563	0	1
Onchain Cost (Modified Normalized)	0.672	0.437	1	0
Liveness (txn data, state update)	7, 60 mins	1, 60 mins	240, 48 mins	59, 60 mins
Liveness	0.025, 1	0, 1	1, 0	0.243, 1
Liveness (Modified Normalized)	0.975, 0	1, 0	0, 1	0.757, 0
Finality Time	168 hrs	152 hrs	0 hrs	3 hrs
Finality Time Normalized	1	0.905	0	0.018
Finality Time (Modified Normalized)	0	0.095	1	0.982
Risk	4	4.5	3.5	3
Risk Normalized	0.667	1	0.333	0
EVM Compatibility	1	1	0	1
Programming Language	1	1	0	1
Stage	0.5	0.5	0.5	0

Table 4.1: Comparison of Layer 2 Solutions Across Key Metrics

4.4.3 Example of Final Score Calculation

Criteria	Weight
Scalability	9
Security	7
Decentralization	6
Cost Efficiency	8
Developer Experience	5

Table 4.2: User Assigned Weights For Evaluation Criteria

In a hypothetical situation, if a user assigns the weights 9 to scalability, 7 to security, 6 to decentralization, 8 to cost efficiency and 5 to developer experience as shown in Table 4.2, then the final score of each protocol would be calculated as shown:

Optimism

- **Scalability:** UOPS = 0.558, Liveness = 0.4875 $\rightarrow$ Avg = 0.523
- **Security:** Risk = 0.667, TVL = 0.197, Finality = 0 $\rightarrow$ Avg = 0.288
- **Decentralization:** Stage = 1 $\rightarrow$ Value = 0.5
- **Cost Efficiency:** 0.672

- **Developer Experience:** EVM = 1, Language = 1 → Avg = 1.0
- **Final Score:** $\frac{(9\times0.4875)+(7\times0.288)+(6\times0.5)+(8\times0.672)+(5\times1)}{5} = 4.02$

Arbitrum

- **Scalability:** UOPS = 1, Liveness = 0.500 → Avg = 0.750
- **Security:** Risk = 1, TVL = 1, Finality = 0.095 → Avg = 0.698
- **Decentralization:** Stage = 1 → Value = 0.5
- **Cost Efficiency:** 0.437
- **Developer Experience:** EVM = 1, Language = 1 → Avg = 1.0
- **Final Score:** $\frac{(9\times0.750)+(7\times0.698)+(6\times0.5)+(8\times0.437)+(5\times1)}{5} = 4.63$

Starknet

- **Scalability:** UOPS = 0.334, Liveness = 0.500 → Avg = 0.41700
- **Security:** Risk = 0.333, TVL = 0.00325, Finality = 1 → Avg = 0.445
- **Decentralization:** Stage = 1 → Value = 0.5
- **Cost Efficiency:** 1.0
- **Developer Experience:** EVM = 0, Language = 0 → Avg = 0.0
- **Final Score:** $\frac{(9\times0.417)+(7\times0.445)+(6\times0.5)+(8\times1)+(5\times0)}{5} = 3.574$

ZKsync Era

- **Scalability:** UOPS = 0, Liveness = 0.379 → Avg = 0.189
- **Security:** Risk = 0, TVL = 0, Finality = 0.982 → Avg = 0.327
- **Decentralization:** Stage = 0 → Value = 0.0
- **Cost Efficiency:** 0.0
- **Developer Experience:** EVM = 1, Language = 1 → Avg = 1.0
- **Final Score:** $\frac{(9\times0.189)+(7\times0.327)+(6\times0)+(8\times0)+(5\times1)}{5} = 1.798$

Protocol	Final Score
Arbitrum	4.630
Optimism	4.020
Starknet	3.574
ZKsync Era	1.798

Table 4.3: Final Summary of Protocol Scores Based on User-Defined Weights

In Table 4.3, we can see the final scores of the four protocols we have selected, which shows that Arbitrum is the most suitable for this particular user based on their requirements, and ZKsync Era is the least suitable.

Chapter 5

Architecture & Protocol Flow

5.1 Architecture

To understand the workflow of our system, we can look at the architecture in Figure 5.1, where the components User, WebApp, Evaluation Engine, Deployment Engine, and each of the protocol's testnets are present. Each of the components will receive certain information, and perform a set of tasks and send that to the next component. We will now describe what the components are.

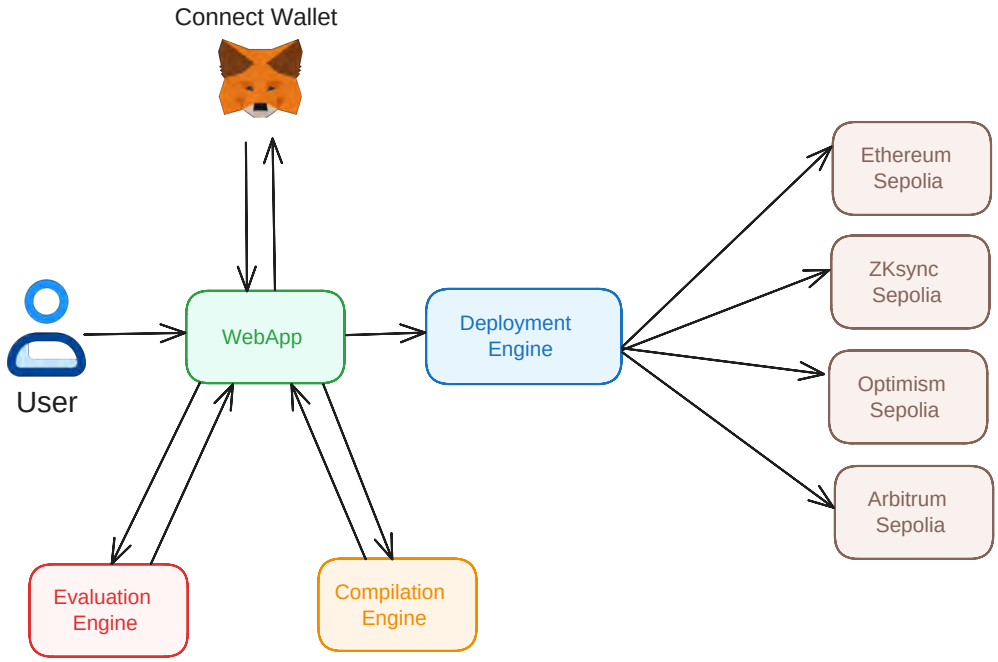

Figure 5.1: Architecture Diagram

I **User:** Users consists of business owners, CEOs or project managers, who are new to Ethereum based dApps and are unsure about which scalability Layer 2 solution will be best for their business.

II **Evaluation Engine:** The evaluation engine fetches the user preferences from the WebApp and assigns weights to various evaluation criteria. Then it calculates the scores for each analysis and returns the total score for each protocol considering the users' priorities.

III **WebApp:** this is going to be where it will be decided which protocol(s) are suitable for the user according to their preferences. After the user submitted their preferences to WebApp, it assigns scores to various deployment options based on factors like cost, security, scalability, and latency requirements. An advanced scoring system applies an evaluation formula to assign scores to various protocols. This system ranks the available options or directly selects the most suitable one. Once the evaluation is complete, the WebApp forwards the collected data and evaluation results to the Deployment Engine for further processing.

IV **Compilation Engine:** The compilation engine is responsible for compiling the smart contract and generating the bytecode and abi of the smart contarct.

V **Deployment Engine:** The sole purpose of the deployment engine is to deploy the user's selected smart contract onto the appropriate protocol (ZKSync, Optimism, or Arbitrum). The deployment engine interface will provide the user with a visual of the options of a single or multiple protocols (will vary according to the user's preferences) based on the results and data that came from the WebApp. With a simple action, users can deploy their smart contracts seamlessly to the chosen protocol(s).

VI **Test Net:** This refers to the test networks of the three Layer 2 protocols. The system will launch the user's smart contract in one or more of these test nets to solidify the system evaluations if the user wishes to deploy.

At first, the user interacts with the web app and enters their preference based on various parameters which determine its heuristic value and security. The user will mark how important each of the criteria is to them (for instance, rate from 1 to 10 how important it is for them to have a higher UOPS), after which, based on the user's preferences, the evaluation engine will calculate the results for each of the protocols from best to worst. The user, after seeing the results, can choose which protocol to use, and based on their choice (ZkSync, Optimism, or Arbitrum), the deployment engine will deploy the users smart contract on that particular Layer 2 protocol.

5.2 Protocol Flow

Next, we used the framework in Figure 5.2, Figure 5.3 and Figure 5.4 to demonstrate a protocol flow. This flow illustrates the interactions between different components of the proposed architecture.

I **Processing User Preference:** The user accesses the web app through user interface and provides their desired preferences. The preferences are collected and compiled into an object, which the web app passes to the evaluation engine for further processing.

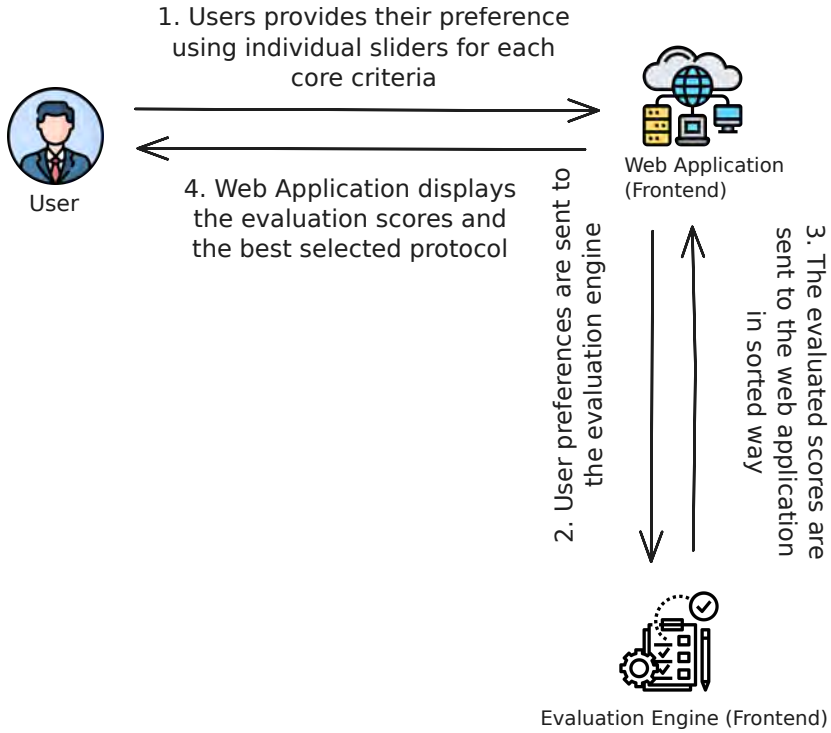

Figure 5.2: Processing User Preference

II **Protocol Evaluation:** The evaluation engine which is a part of the frontend of the web app receives the object containing user preference. Then using the pre-stored protocol heuristic values, data from user preference objects, and the framework formulas this engine calculates evaluation score for each protocol. Finally, the evaluation engine returns the calculated evaluations score to the web app and the web app presents this information to the user.

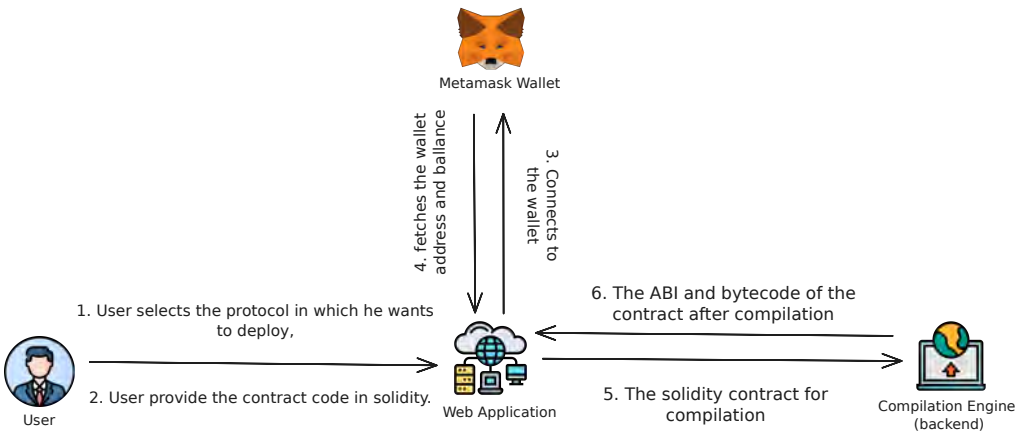

Figure 5.3: Contract Deployment Initialization

III **Contract Deployment Initialization:** After receiving the evaluation, the users can deploy their contract in one or more supported Layer 2 protocols. If the user wishes to do so, the web app provides them interfaces to connect their Metamask wallet and provide their solidity code. The web app then sends the solidity smart contract to the compilation engine. The compilation engine return the ABI and the bytecode of the smart contract to the web application.

IV **Contract Deployment Preparation:** After receiving required data from the compilation engine the web app sends selected protocol info to the deployment engine. The deployment engine retrieves the required RPC url and chain id from the protocol. It then asks the user to give their digital signature for contract deployment. Once it receives the users digital signature it deploys the contract in the selected protocol.

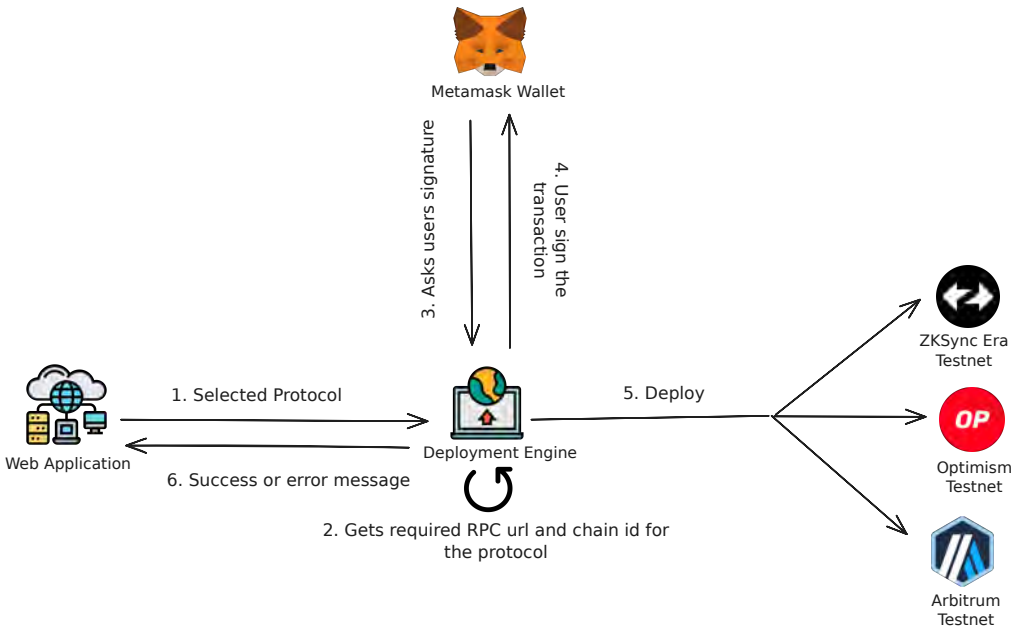

Figure 5.4: Contract Deployment

V **Contract Deployment and Status Feedback:** If the deployment engine is able to successfully deploy the smart contact it returns a success message along with the address of the launched smart contract. If it fails to launch the smart contract then it returns an error message. The appropriate message is displayed to the user on the user interface.

Chapter 6

System Implementation

This chapter provides an overview of the system implementation for the Ethereum Layer-2 Evaluation Framework. The system is divided into three main components: the frontend interface, the backend service, and the deployment infrastructure. Together, these elements enable a seamless user experience for evaluating and deploying smart contracts on different Layer-2 networks. Even though we had used alchemy and foundry in our preliminary system implementation, we have decided not to continue with those, as foundry requires containers and the use of dockers that requires higher engineering skills and expertise that we lack, and alchemy is removed because we are handling the deployment process by using the RPC URL of the networks and Wagmi [35] instead of using a third party service provider.

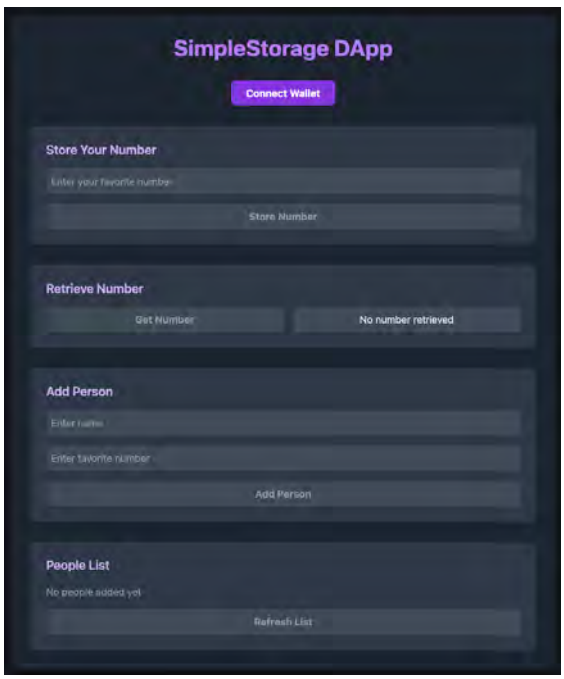

Figure 6.1: Wallet Not Connected

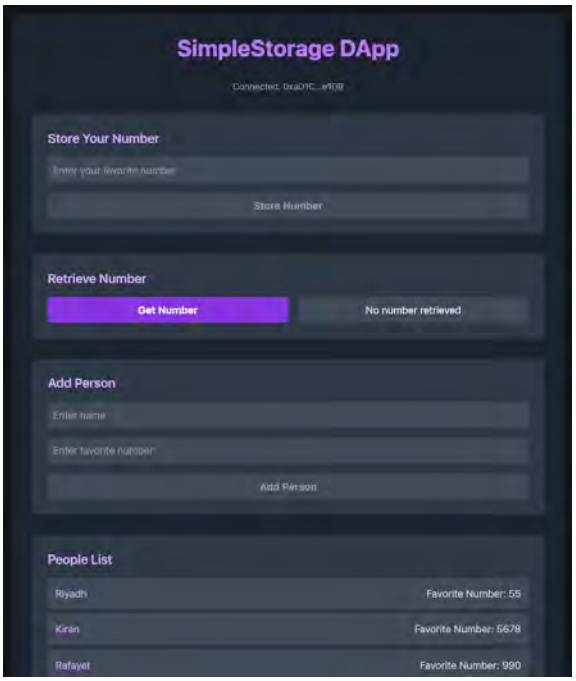

Figure 6.2: Wallet Connected

6.1 Preliminary System Implementation

As our research revolves around a framework, it was a necessary first step to gain hands-on experience with blockchain development, which is why we conducted a preliminary system deployment as shown in Figure 6.2 and Figure 6.1. This was a small-scale implementation which helped us understand the basics of development in blockchain.

We used Foundry [62] (Ethereum Development Framework) and Solidity [64] (Programming Language for Ethereum) to gain practical experience in blockchain development. We developed a basic smart contract in Solidity and utilized the Foundry's local testnet built for ZKsync Era to deploy the smart contract several times, simulating various transactions to evaluate the capabilities of it.

To ensure a smooth deployment process we integrated Alchemy [54], a blockchain infrastructure provider, to access high performance Ethereum nodes to connect with the testnet of ZKsync [63]. This approach was an essential step for handling network communication, improving transaction reliability while ensuring efficient contract deployment. Alchemy's [54] compatibility with ZKsync [63] allowed us to bypass some of the complexities of manual node management while enhancing our system's responsiveness and scalability.

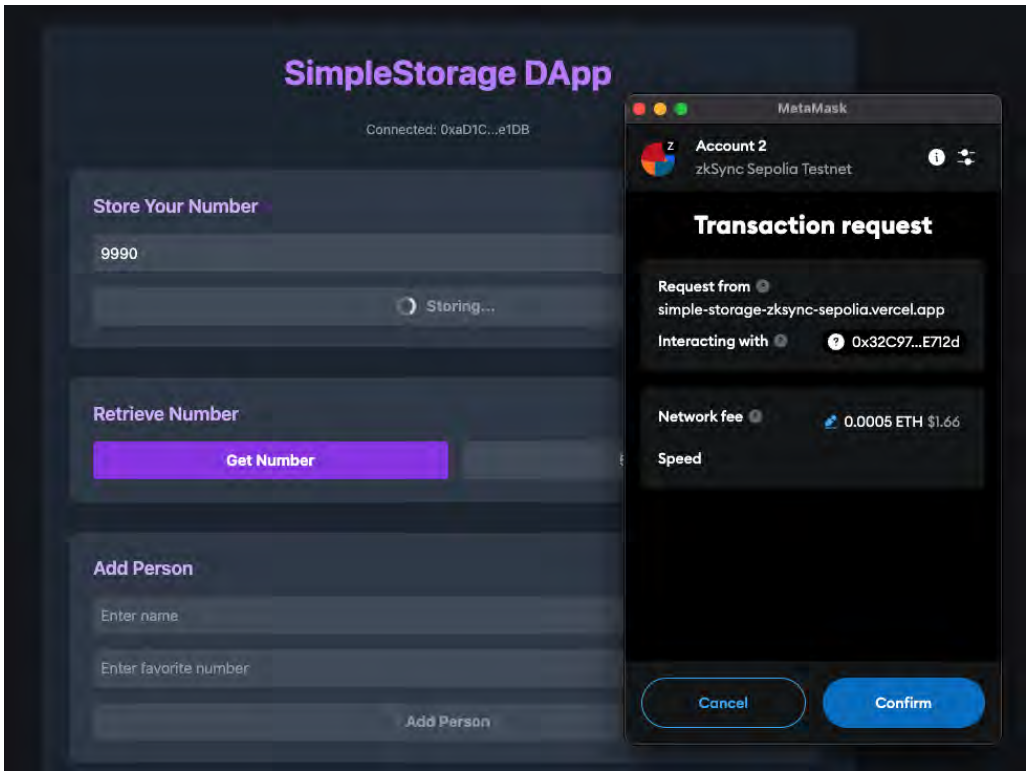

Figure 6.3: Zksync Transaction Confirmation

This was a very simple implementation of a solidity smart contract, we called it 'SimpleStorage DApp'. As the purpose was not to build a robust application rather to get ourselves familiar with the technologies we are going to deal with, we chose

a very simple implementation. You need to connect your wallet before you can interact with the app. As you can see in Figure 6.1 as we were not connected with a wallet the buttons were disabled and People List were also hidden. As soon as we connected our wallet you can see in Figure 6.2 the button were enabled, and we could start interacting with the app.

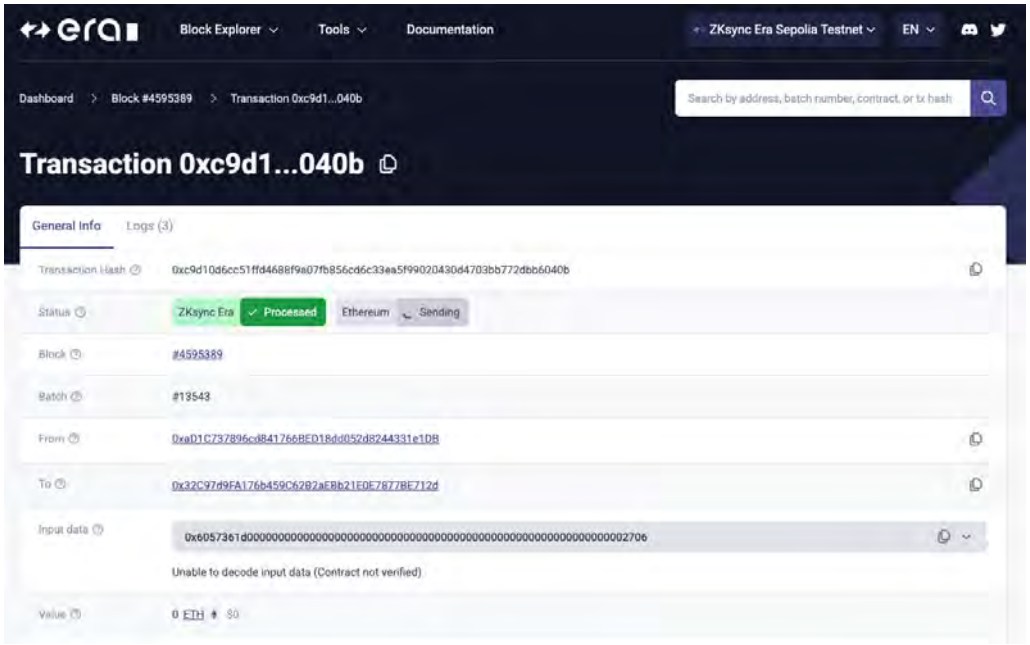

Figure 6.4: Transaction in ZKsync Block Explorer

In the app we have a way to store a number and retrive the number on chain. Then we can add a person with their favorite number on chain as well. To confirm a transaction our wallet will ask us to confirm and sign a transaction as shown in Figure 6.3. Also we can see our transations in ZKsync block explorer [59] as we can see in Figure 6.4.

6.2 Final System Implementation

6.2.1 System Frontend

The frontend serves as the entry point for users, built using the Next.js [49] framework. At the very first page of our app (Figure 6.5) the user sees a slider like form to provide their preference about different aspect of the Layer 2 protocols. We have used Tailwind CSS [41] to ensure a clean and beautiful user interface. TypeScript [42] is used throughout the application to enforce type safety and improve maintainability. For blockchain interactions, we use Wagmi [35] for wallet connectivity and Viem [34] for managing multiple chains with strong type safety. The frontend is deployed using Vercel [37], which provides continuous deployment, global CDN delivery, and scalability.

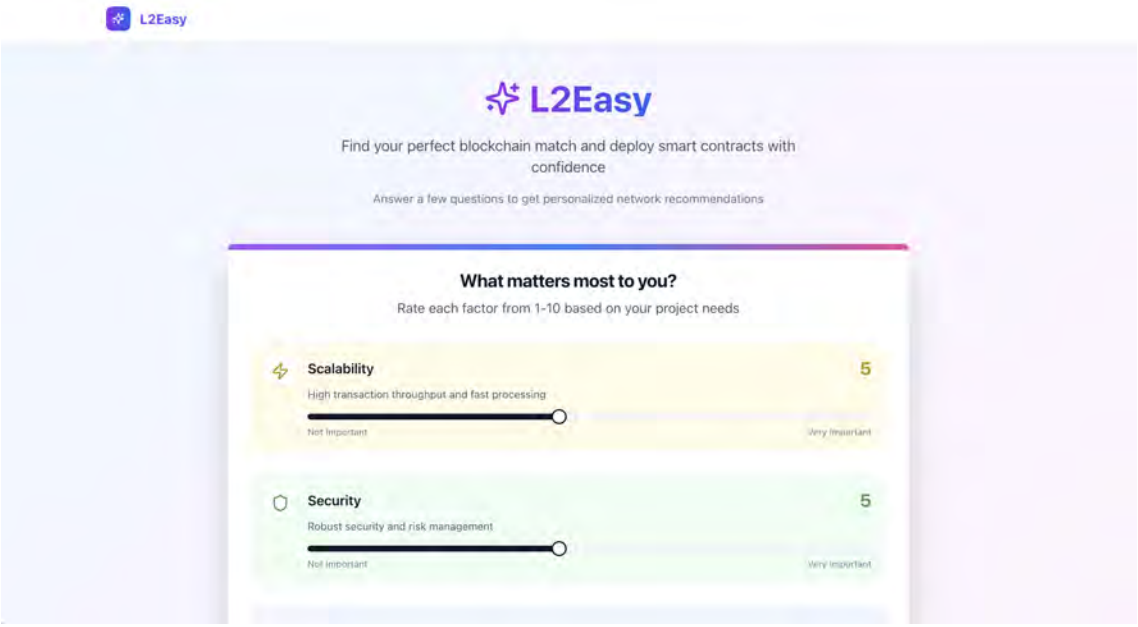

Figure 6.5: L2Easy Home Page

6.2.2 Evaluation Engine

The evaluation engine allows users to rate key criteria such as scalability, security, decentralization, cost efficiency, and developer experience using interactive sliders. These preferences are used in the frontend for weighted score computation. The results are returned as shown in Figure 6.6. They are also ranked from best to worst in the same page near the bottom.

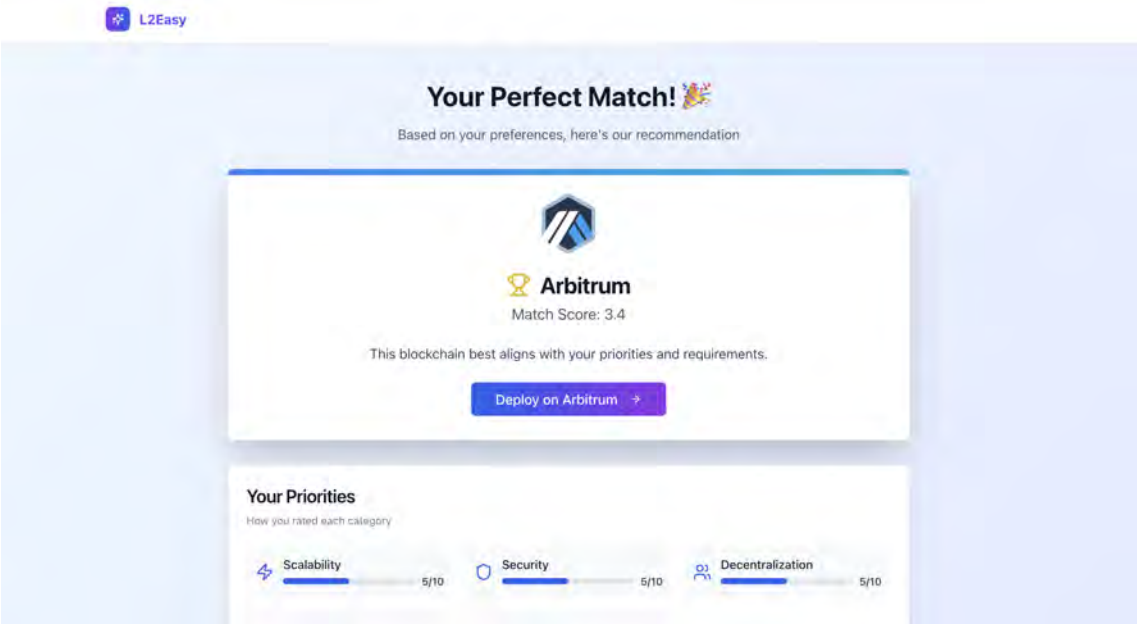

Figure 6.6: L2Easy Result Page

6.2.2.1 Pseudocode

This a code example in Javascript that shows the implementation of the weighted scoring system that **L2Easy** uses. This particular algorithm can be used for any Layer 2 protocol that is currently compatible.

Inputs:

- `entity_data`: A dictionary where each key is an entity name and each value is a dictionary of metric values.
- `user_weights`: A dictionary with metric categories as keys and user-defined numeric weights as values.
- `normalization_factor`: A scalar value used to normalize the final score (e.g., 5 for a 1–10 range).

```
function calculateFinalScores(protocolData, userWeights) {
  const stageMapping = { 0: 0.0, 1: 0.5, 2: 1.0 };
  const finalScores = {};

  for (const protocolName in protocolData) {
    const metrics = protocolData[protocolName];

    const scalability = (metrics.TPS + metrics.Liveness) / 2;
    const security = (metrics.Risk + metrics.TVL + metrics.Finality
       ) / 3;
    const decentralization = stageMapping[metrics.Stage];
    const costEfficiency = metrics.Cost;
    const devExperience = (metrics.EVM + metrics.Language) / 2;

    const weightedScore =
      (userWeights["Scalability"] * scalability) +
      (userWeights["Security"] * security) +
      (userWeights["Decentralization"] * decentralization) +
      (userWeights["Cost Efficiency"] * costEfficiency) +
      (userWeights["Dev Experience"] * devExperience);

    const finalScore = Number((weightedScore / 5).toFixed(4));
    finalScores[protocolName] = finalScore;
  }

  return finalScores;
}
```

Listing 6.1: Weighted Scoring Algorithm

6.2.3 Deployment Engine

The deployment engine facilitates smart contract deployment after getting the bytecode and abi from the compilation engine on section 6.2.5. The user can provide a

smart contract and get the bytecode and ABI of the contract using our compilation engine. After that they can just hit deploy and Wagmi [35] will hash the bytecode, ABI and their account address and request for a transaction in the selected network. Once the user signs the transaction using their wallet the transaction becomes live to the network and the contract address is returned to the user interface if successful.

6.2.4 System Backend

The backend is only responsible for the compilation of the smart contract and returning the bytecode and ABI of the contract to the frontend. It is hosted on Render [46] and built using Node.js [36], Express [33], with REST APIs to interact with the frontend.

6.2.5 Compilation Engine

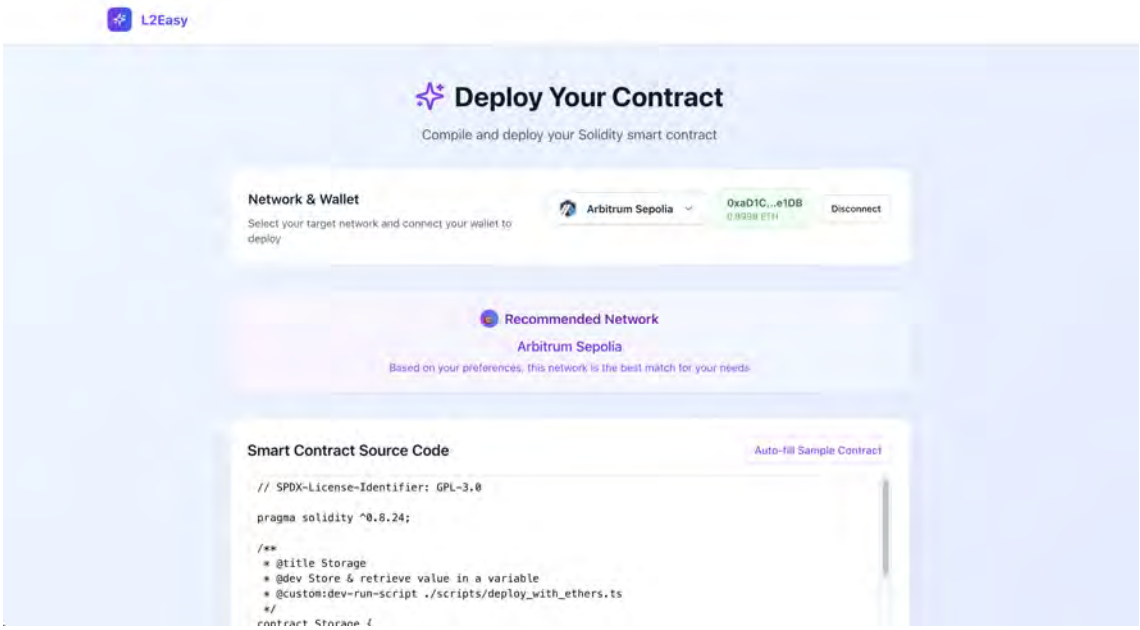

Figure 6.7: L2Easy Deploy Page

The compilation engine, as shown in Figure 6.7, takes in the Solidity [64] code from the frontend and compiles it using the solc compiler. The resulting ABI and bytecode are sent back to the frontend for deployment. This work is done in a separate backend service because sometimes the contract can be heavy in size and it becomes harder for the client to compile the contract.

6.2.6 System Deployment

The complete system is deployed using modern cloud-based platforms:

- The frontend is deployed on Vercel [37], optimized for Next.js [49] applications with CI/CD and edge delivery.
- The backend is deployed on Render, providing scalable server-side processing for compilation of the smart contracts.

The process ends with a success or error response returned to the frontends shown in Figure 6.8.

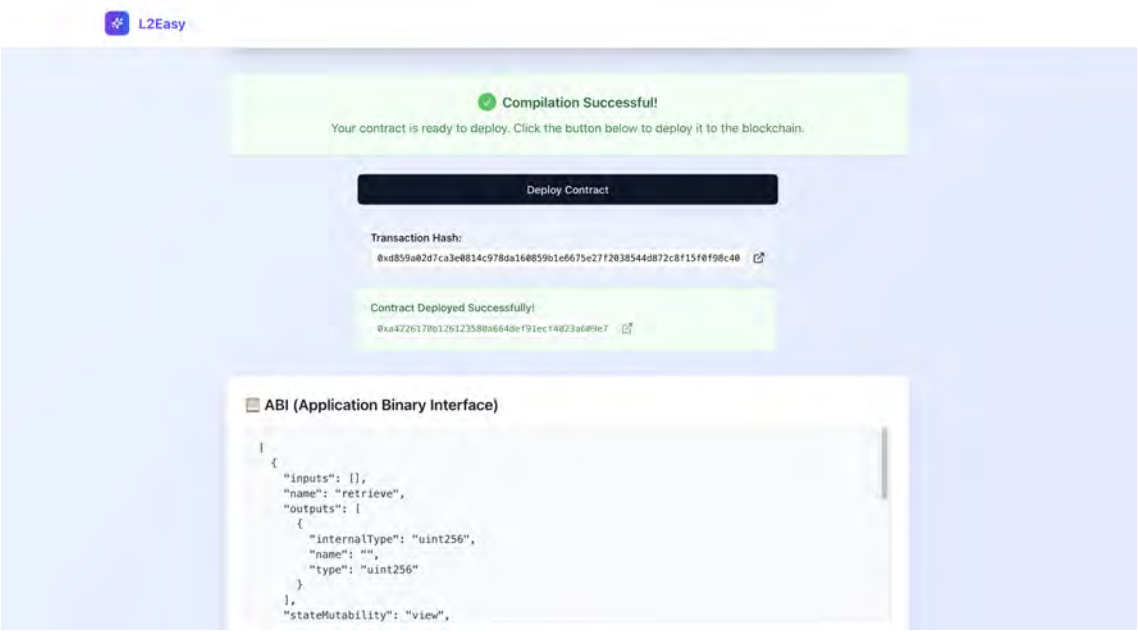

Figure 6.8: L2Easy Contract Deployment Successful

Chapter 7

Discussion

As Layer 2 protocols show potential to make blockchain technology based services more scalable, these are being adopted at a rapid rate. Also, the number of these Layer 2 protocols are increasing rapidly with each passing day. In the course of our research we have realized that the Layer 2 protocols vary significantly despite being built over the same fundamentals due to their internal structure. Moreover, very few research and industrial studies have tried to compare these Layer 2 protocols and none tried to do so on a numerical basis. For any service provider, it is not only a hassle but also somewhat impractical to survey all blockchain Layer 2 protocols to find the one that suits their needs the most. This article presents a blockchain Layer 2 protocol evaluation framework that not only compares these protocols based on user preference input, but also takes a numerical evaluation approach. The evaluation protocol is not coupled to any specific type of Layer 2 protocols and can be used to evaluate any Ethereum based Layer 2 protocol. This is possible because the framework utilizes heuristics that are common across all Ethereum based and perhaps many other Layer 2 protocols.

7.1 Requirement Analysis

In this section we will discuss how we fulfilled our functional and non functional requirements by implementing **L2Easy**. Our first functional requirement was fulfilled through the very first page users sees after landing in our system, see Figure 6.5. The second and third functional requirements were fulfilled through the evaluation engine of our system. The remaining functional requirements were fulfilled through the compilation engine and deployment engine of our system. And all the non functional requirements were also fulfilled by maintaining consistent design throughout the app that is easy to use and understand for a normal user. And by using cutting edge frontend technologies to provide users with the best in class experience.

7.2 Research Objective Analysis

Our **RO1** was to thoroughly examine the latest research and systems within the scope of Ethereum Layer 2 solutions, with a specific focus on scalability, security, and interoperability. **RO1** was addressed in Chapter 3, by conducting a detailed literature review of contemporary blockchain systems, focused on existing Layer 2 technologies. Additionally, a system-level review of four prominent Layer 2 protocols Arbitrum, Optimism, Starknet, and ZKsync Era was performed to understand their working mechanism and technical characteristics in relation to the identified focus areas.

RO2 of our research was to analyze user preferences, heuristic indicators, and security aspects of the selected Layer 2 protocols. This was achieved in Chapter 4 by first identifying five core criteria: decentralization, transaction throughput, finality time, data availability, and cost-efficiency from fundamental blockchain properties. Heuristic data was extracted from L2BEAT, a reputable analytics platform for Layer 2 networks, and subsequently mapped to the defined core criteria. This mapping allowed for a structured evaluation of each protocol based on both technical and user-centric metrics.

Our **RO3** was to create an architecture for the Layer 2 evaluation framework that satisfies key functional and non-functional requirements. These requirements included performance scalability, security resilience, and interoperability with Ethereum Layer 1. This objective was achieved in Chapter 5 by proposing a modular framework architecture composed of evaluation components for protocol data ingestion, numerical evaluation, score visualization, and contract deployment. The architectural design was validated against the requirements to ensure that it could robustly support the evaluation of both existing and emerging Layer 2 solutions.

And finally, our **RO4** was to implement a prototype system that embodies the proposed evaluation framework and to assess its feasibility and usability. This was realized through the development of a web-based application that evaluates Layer 2 solutions based on the proposed framework and extracted data from L2BEAT, which is done in Chapter 6. The system was tested with various types of smart contracts, from small to large ones with different protocols. They performed smoothly in all the tests without breaking.

7.3 Limitations

- Static Heuristic Inputs: The framework relies on manually collected heuristic data (e.g., UOPS, cost, TVS, risk), which are point-in-time values. These metrics may become out-dated quickly due to the fast-evolving nature of Layer 2 protocols.
- Subjective Weight Dependency: The scoring output is directly influenced by the user-defined weights across criteria. While this allows personalization, it

introduces subjectivity, which limits the framework's applicability for standardized or policy-level decisions.

- Simplified Abstraction of Complex Properties: Concepts such as decentralization and security are represented by simplified metrics like "Stage" or "Finality," which may not fully capture their multi-faceted nature in practical deployments.
- Limited Protocol Coverage: Only four Layer 2 protocols - Optimism, Arbitrum, Starknet, and ZKsync Era - were evaluated. Other emerging solutions such as Scroll, Base, and Linea were not included, which restricts the comprehensiveness of the analysis.
- EVM-Centric Developer Experience Scoring: Protocols that do not fully support the EVM (e.g., Starknet with Cairo) may be disadvantaged in developer experience scoring, despite offering novel and performant architectures.
- Hard to find people who are familiar with Layer 2: We could not perform any user testing with our web app because we could not find enough people who could give us constructive feedback regarding the usecase of our framework on Layer 2.

7.4 Challenges

- Data Inconsistency Across Protocols: Each protocol documents metrics like UOPS, TVL, or cost differently, often lacking a standardized or transparent methodology. This created challenges in ensuring fair and consistent normalization across platforms.
- System Implementation Barriers: Deploying and testing contracts across different Layer 2 presented integration issues, including incompatibility in development environments, varying deployment flows, and lack of unified SDKs or documentation.
- Absence of Real-Time Integration: The scoring framework does not fetch or reflect real-time data. Incorporating APIs from L2BEAT or similar sources would require significant backend work and maintenance.
- Difficult to manage Eth or test Eth: There was no easy way to get test Eth which requires us to have actual eth, which is also difficult to have as we did not have enough funds or the means to keep eth in our wallets.
- Reinventing the wheel: The open-source code of deployment service providers like Remix was difficult to interpret and mimic, so we had to improvise and come up with a better way to help our users to deploy contracts directly from our webapp.

7.5 Future Work

As of now the proposed evaluation framework and its accompanying web-based application support analysis of only the four selected Layer 2 solutions Arbitrum, Optimism, Starknet, and ZKsync Era due to its reliance on static data management. This approach was adopted due to the inadequate support for efficient and comprehensive API access during the development phase. We intend to increase the robustness and scalability of the system by integrating dynamic data management through real-time API connections, particularly leveraging the L2BEAT API. This integration will allow for automated data updates, improved responsiveness to changing protocol metrics, and reduced manual intervention.

Another limitation of the system is the inability to deploy provided solidity contract on Layer 2 protocols that has a different native language. The system currently cannot deploy smart contracts on Starknet as its smart contracts are written in Cairo. We could not overcome the limitation due to not finding any services that translate a solidity program to Cairo. In future, we intend to make the system capable of deploying contracts on platforms that uses languages other than solidity.

Moreover, the current compiler version only supports one version, but the expansion of versions could be helpful in compiling a larger scale of smart contracts. In addition to supporting complex smart contract compilation with dependencies such as libraries and multiple files, auto verification of the smart contract the scope for the compilation section can be enhanced as well.

One other notable limitation is in the current scope of core evaluation criteria, which is constrained by the availability of heuristics that can be quantified and mapped to measurable indicators. Expanding the set of evaluation criteria is essential in order to improve the comprehensiveness of the framework. This expansion will involve taking additional heuristics and metrics from reliable sources beyond L2BEAT, such as official documentation, whitepapers, GitHub repositories, and direct communication with protocol teams. These enhancements aim to increase the reliability and applicability of the framework across a broader set of Layer 2 protocols by supporting more thorough and multidimensional evaluation.

In addition to this, we intend to test the system with proper users, as this system cannot be tested without prior technical knowledge of Blockchain and basic knowledge of Ethereum Layer 2. We also need access to a decent amount of test Ether to give to the users for testing purposes. The platform involves advanced functionalities such as wallet integration, smart contract deployment, and protocol-level configuration requiring users to possess a baseline familiarity with blockchain development practices. Without testing conducted by professional personnel, the usability cannot be evaluated accurately.

Chapter 8

Conclusion

In conclusion, our motivation for building a Layer 2 evaluation and proposal system emerged from the fact that Layer 2 solutions are the most promising approach of the current age at solving scalability issues of Ethereum, and other blockchain systems. In the pre thesis 1, we examined the system architecture of five Layer 2 protocols and reviewed related academic literature. By doing so we were able to meet our **RO#1** of exploring and analyzing the current state of the art of Layer 2 solutions of Ethereum. In pre thesis 2, we completed user preference, heuristic, and security analysis of our selected protocols which addresses our **RO#2**. We have also proposed an architecture and conducted both requirement analyses of the proposed system which partially addressed our **RO#3** during pre thesis 2. In addition to that, with the successful design, development and deployment of the **L2Easy** application, we have now fully addressed **RO#3** and **RO#4**. The application enables users to compare Layer 2 protocols based on weighted preferences, gain actionable insights, and even deploy smart contracts to selected testnets. Looking ahead, we plan to refine protocol scoring after collecting user feedback while testing and integrate real time performance metrics further enhancing **L2Easy** as a powerful decision support tool for developers, researchers, and blockchain users worldwide. Future work may involve extending this system further by incorporating additional protocols such as sidechains, app-specific rollups and emerging hybrid models into the system addressing the limitations encountered in this research.

Bibliography

[1] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system." [Online]. Available: https://bitcoin.org/bitcoin.pdf. (2008).

[2] V. Buterin, *Sharding faq*, Accessed: 2025-05-29, 2017. [Online]. Available: https://vitalik.eth.limo/general/2017/12/31/sharding_faq.html#this-sounds-like-theres-some-kind-of-scalability-trilemma-at-play-what-is-this-trilemma-and-can-we-break-through-it.

[3] D. Guegan, *Public blockchain versus private blockchain*, HAL Archives, Research Report, Available: `halshs-01524440`, 2017.

[4] A. Chauhan, O. P. Malviya, M. Verma, and T. S. Mor, "Blockchain and scalability," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Jul. 2018. DOI: 10.1109/qrs-c.2018.00034.

[5] s. Sarmah, "Understanding blockchain technology," *Scientific Academic Publishing*, vol. 8, pp. 23–29, Aug. 2018. DOI: 10.5923/j.computer.20180802.02.

[6] Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018. DOI: 10.1504/IJWGS.2018.10009118.

[7] W.-M. Lee, *Beginning Ethereum Smart Contracts Programming*. Berkeley, CA: Apress, 2019. DOI: 10.1007/978-1-4842-5086-0.

[8] StarkWare Blog. "Zk-rollups vs optimistic rollups." [Online]. Available: https://starkware.co/blog/zk-rollups-explained/zk-rollups-vs-optimistic-rollups/. [Accessed: Oct. 12, 2024]. (May 2019).

[9] M. S. Uddin, *Understanding blockchain technology*, Accessed: 2025-05-29, 2019. [Online]. Available: https://d1wqtxts1xzle7.cloudfront.net/60715489/Understanding_Blockchain_Technology20190926-26770-147v9qi-libre.pdf.

[10] V. Buterin, *A rollup-centric ethereum roadmap*, Accessed: 2025-06-11, 2020. [Online]. Available: https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698.

[11] S. Ferdous, M. A. Chowdhury, M. O. Hoque, and A. Colman, "Blockchain consensus algorithms: A survey," *arXiv (Cornell University)*, Jan. 2020. DOI: 10.48550/arxiv.2001.07091.

[12] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE Access*, vol. 8, pp. 125 244–125 262, 2020. DOI: 10.1109/access.2020.3007251.

[13] C. Komalavalli, D. Saxena, and C. Laroiya, "Overview of blockchain technology concepts," in *Blockchain Technology: Concepts and Applications*, S. Rani, N. Dey, and A. Joshi, Eds., Elsevier, 2020, pp. 349–371, ISBN: 9780128198162. DOI: 10.1016/B978-0-12-819816-2.00014-9.

[14] M. S. Ferdous, M. J. M. Chowdhury, and M. A. Hoque, "A survey of consensus algorithms in public blockchain systems for crypto-currencies," *Journal of Network and Computer Applications*, vol. 182, p. 103 035, May 2021. DOI: 10.1016/j.jnca.2021.103035.

[15] C. Sguanci, R. Spatafora, and A. M. Vergani, "Layer 2 blockchain scaling: A survey," Jun. 2021, https://arxiv.org/pdf/2107.10881.

[16] N. Chemaya and D. Liu. "Cost of security of layer 2 network: Evidence from polygon network." [Online]. Available: https://ssrn.com/abstract=4119827. (May 2022).

[17] L. Donno, "Optimistic and validity rollups: Analysis and comparison between optimism and starknet," *arXiv*, Oct. 2022. DOI: 10.48550/arxiv.2210.16610. [Online]. Available: https://doi.org/10.48550/arxiv.2210.16610.

[18] A. Gangwal, H. R. Gangavalli, and A. Thirupathi, "A survey of layer-two blockchain protocols," *Journal of Network and Computer Applications*, vol. 209, p. 103 539, Nov. 2022. DOI: 10.1016/j.jnca.2022.103539.

[19] T. Lavaur, J. Lacan, and C. P. C. Chanel, "Enabling blockchain services for ioe with zk-rollups," *Sensors*, vol. 22, no. 17, 2022. DOI: 10.3390/s22176493.

[20] P. Purwono, A. Ma'arif, W. Rahmaniar, Q. M. ul Haq, D. Herjuno, and M. Naseer, "Blockchain technology," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 8, no. 2, p. 199, Jul. 2022. DOI: 10.26555/jiteki.v8i2.24327.

[21] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93 039–93 054, Jan. 2022. DOI: 10.1109/access.2022.3200051.

[22] L. Zhang, X. Wei, Y. Liu, Y. Yang, Z. Yu, and M. Guo, "Evm-compatible blockchain: Scaling smart contracts for the future," *arXiv preprint arXiv:2208.10269*, 2022, Accessed: 2025-05-29. [Online]. Available: https://arxiv.org/pdf/2208.10269.

[23] Crypto.news. "Developer shortage in the crypto space is where the real problem lies." Accessed: 2025-05-29. (2023), [Online]. Available: https://crypto.news/developer-shortage-in-the-crypto-space-is-where-the-real-problem-lies-opinion/.

[24] Hyperledger Fabric Contributors, *Blockchain concepts — hyperledger fabric documentation*, https://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html, Accessed: 2025-04-21, 2023.

[25] ilia. "Simple decentralized protocol proposal." [Online]. Available: https://community.starknet.io/t/simple-decentralized-protocol-proposal/99693.. [Accessed: Oct. 14, 2024]. (2023).

[26] R. Neiheiser, G. Inácio, L. Rech, C. Montez, M. Matos, and L. Rodrigues, "Practical limitations of ethereum's layer-2," *IEEE Access*, vol. 11, pp. 8651–8662, Jan. 2023. DOI: 10.1109/access.2023.3237897.

[27] R3, *Corda: The leading permissioned blockchain platform*, https://r3.com/corda/, Accessed: 2025-04-21, 2023.

[28] I. Roșca, A.-I. Butnaru, and E. Simion, *Security of ethereum layer 2s*, https://eprint.iacr.org/2023/124, Cryptology ePrint Archive, 2023. [Online]. Available: https://eprint.iacr.org/2023/124 (visited on 10/14/2024).

[29] thirdweb Blog. "What is layer 2 blockchain? an introduction to layer 2 scaling solutions." [Online]. Available: https://blog.thirdweb.com/what-is-layer-2-blockchain/. [Accessed: Oct. 12, 2024]. (2023).

[30] S. Yoo, "Comparative analysis of blockchain trilemma," *International Journal of Advanced Smart Convergence*, vol. 12, no. 1, pp. 41–52, 2023. DOI: 10.7236/IJASC.2023.12.1.41.

[31] Y. I. Alzoubi and A. Mishra, "Techniques to alleviate blockchain bloat: Potentials, challenges, and recommendations," *Computers & Electrical Engineering*, vol. 116, p. 109 216, May 2024. DOI: 10.1016/j.compeleceng.2024.109216.

[32] M. Capretto, M. Ceresa, A. F. Anta, P. Moreno-Sánchez, and C. Sánchez, "Fast and secure decentralized optimistic rollups using setchain," *arXiv (Cornell University)*, Jun. 2024. DOI: 10.48550/arxiv.2406.02316.

[33] E. Contributors, *Express - node.js web application framework*, https://expressjs.com/, Accessed: 2025-06-04, 2024.

[34] V. Contributors, *Viem: Type-safe ethereum development*, https://viem.sh/, Accessed: 2025-06-04, 2024.

[35] W. Contributors, *Wagmi: React hooks for ethereum*, https://wagmi.sh/, Accessed: 2025-06-04, 2024.

[36] O. Foundation, *Node.js*, https://nodejs.org/en, Accessed: 2025-06-04, 2024.

[37] V. Inc., *Vercel: Develop. preview. ship.* https://vercel.com/, Accessed: 2025-06-04, 2024.

[38] H. Ko and S. Han, "Tps analysis, performance indicator of public blockchain scalability," *Journal of Information Processing Systems*, vol. 20, no. 1, pp. 85–92, 2024. DOI: 10.3745/JIPS.04.0304.

[39] M. Kolombet. "Multi-prover ethereum layer 2 rollups." [Online]. Available: https://epublications.vu.lt/object/elaba:204759866/. (2024).

[40] L2BEAT. "Scaling summary." [Online]. Available: https://l2beat.com/scaling/summary. [Accessed: Oct. 16, 2024]. (2024).

[41] T. Labs, *Tailwind css*, https://tailwindcss.com/, Accessed: 2025-06-04, 2024.

[42] Microsoft, *Typescript: Javascript with syntax for types*, https://www.typescriptlang.org/, Accessed: 2025-06-04, 2024.

[43] Optimism Documentation. "Rollup protocol overview." [Online]. Available: https://docs.optimism.io/stack/protocol/rollup/overview. [Accessed: Oct. 12, 2024]. (2024).

[44] Optimism Documentation. "Welcome to the optimism docs." [Online]. Available: https://docs.optimism.io/. (2024), (visited on 10/12/2024).

[45] G. A. F. Rebello, G. F. Camilo, L. A. C. de Souza, *et al.*, "A survey on blockchain scalability: From hardware to layer-two protocols," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 4, pp. 2411–2452, 2024. DOI: 10.1109/COMST.2024.3376252. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10468547/.

[46] I. Render, *Render: Cloud hosting for developers*, https://render.com/, Accessed: 2025-06-04, 2024.

[47] U. Saif, Y. Alshamrani, N. Alghamdi, A. Alarifi, and B. Alturki, "A survey on data availability in layer 2 blockchain rollups: Open challenges and future improvements," *Future Internet*, vol. 16, no. 9, p. 315, 2024. DOI: 10.3390/fi16090315. [Online]. Available: https://www.mdpi.com/1999-5903/16/9/315.

[48] H. Song, Z. Qu, and Y. Wei, "Advancing blockchain scalability: An introduction to layer 1 and layer 2 solutions," *arXiv.org*, 2024, [Online]. Available: https://arxiv.org/abs/2406.13855.

[49] Vercel, *Next.js: The React Framework*, https://nextjs.org/, Accessed: 2025-06-04, 2024.

[50] Zeeve. "Is tps misleading for rollups? here's why uops or gas/sec are better." Accessed: 2025-06-16. (2024), [Online]. Available: https://www.zeeve.io/blog/is-tps-misleading-for-rollups-heres-why-uops-or-gas-sec-are-better/.

[51] zkSync Documentation. "Blocks." [Online]. Available: https://docs.zksync.io/zk-stack/concepts/blocks. [Accessed: Oct. 13, 2024]. (2024).

[52] zkSync Documentation. "Transaction lifecycle." [Online]. Available: https://docs.zksync.io/zk-stack/concepts/transaction-lifecycle. [Accessed: Oct. 12, 2024]. (2024).

[53] zkSync Documentation. "Zksync stack." [Online]. Available: https://docs.zksync.io/zk-stack. [Accessed: Oct. 12, 2024]. (2024).

[54] Alchemy. "Alchemy: Blockchain development platform." (2025), [Online]. Available: https://www.alchemy.com/ (visited on 01/20/2025).

[55] Arbitrum. "Arbitrum: Layer 2 scaling for ethereum." (2025), [Online]. Available: https://arbitrum.io/ (visited on 01/20/2025).

[56] Ethereum Foundation, *Layer 2: Learn about layer 2 scaling solutions*, Accessed: 2025-05-29, 2025. [Online]. Available: https://ethereum.org/en/layer-2/learn/.

[57] D. Ishii, "Privacy in ethereum: Challenges and open research problem," *ResearchGate*, 2025, Available at https://www.researchgate.net/publication/388844053.

[58] A. A. Makhdomi and I. A. Gillani, "Decentralized ridesharing: Overcoming scalability issues with layer 2 solution," in *Proceedings of the 26th International Conference on Distributed Computing and Networking (ICDCN 2025)*, ACM, 2025. DOI: 10.1145/3700838.3703682.

[59] zkSync, *Zksync sepolia explorer*, Online, Accessed: Feb. 1, 2025, 2025. [Online]. Available: https://sepolia.explorer.zksync.io/.

[60] Ethereum.org, *Home*, https://ethereum.org, Accessed: 2024-10-16.

[61] ethereum.org. “Zero-knowledge proofs.” [Online]. Available: https://ethereum.org/en/zero-knowledge-proofs. [Accessed: Oct. 12, 2024]. ().

[62] Foundry Project, *Foundry book: Ethereum development toolchain documentation*. [Online]. Available: https://book.getfoundry.sh/.

[63] Matter Labs, *Zksync: Scaling ethereum*. [Online]. Available: https://zksync.io/.

[64] Solidity Developers, *Solidity — ethereum's smart contract programming language*. [Online]. Available: https://soliditylang.org/.

[65] zkSync Documentation. “Build on zksync.” [Online]. Available: https://docs.zksync.io/build. [Accessed: Oct. 12, 2024]. ().

[66] zkSync Documentation. “Finality.” [Online]. Available: https://docs.zksync.io/zk-stack/concepts/finality. [Accessed: Oct. 13, 2024]. ().

[67] zkSync Mirror Blog. “Zksync mirror article.” [Online]. Available: https://zksync.mirror.xyz/BqdsMuLluf6AlWBgWOKoa587eQcFZq20zTf7dYblxsU. [Accessed: Oct. 12, 2024]. ().