BVB Campus, Vidyanagar, Hubballi – 580031, Karnataka, INDIA.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Mini Project Report

On

Blockchain-Powered Decentralized DNS for Enhanced Security

submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

Karthik K P	01fe22bcs183
Sushmita Math	01fe22bcs184
Harshitaa Katwa	01fe22bcs257
Chinmay Avaradi	01fe22bcs306

Under the guidance of

Jayalaxmi G N

School of Computer Science and Engineering

KLE Technological University, Hubballi

KLE Technological University
Creating Value
Leveraging Knowledge
BVB Campus, Vidyanagar, Hubballi – 580031, Karnataka, INDIA.

2024-2025

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2024-25

CERTIFICATE

This is to certify that project entitled "Blockchain-Powered Decentralized DNS for Enhanced Security " is a bonafied work carried out by the student team Karthik K P - 01fe22bcs183 , Sushmita Math-01fe22bcs184 , Harshitaa Katwa - 01fe22bcs257 , Chinmay Avaradi - 01fe22bcs306 in partial fulfillment of the completion of 5th semester B. E. course during the year 2024 – 2025. The project report has been approved as it satisfies the academic requirement with respect to the project work prescribed for the above said course.

Guide name:
Jayalaxmi G N

Head, SoCSE
Dr. Vijayalakshmi.M.

External Viva-Voce

Name of the examiners Signature with date

1 _____ _____

2 _____ _____

ABSTRACT

The Domain Name System (DNS) is essential for translating human-readable domain names into machine-readable IP addresses, but its centralized nature exposes it to failures, censorship, and cyberattacks. To address these vulnerabilities, this research proposes a blockchain-based decentralized DNS (dDNS) system that enhances security, transparency, and user control. Built on the Ethereum blockchain using Remix IDE and MetaMask, the system employs smart contracts for secure domain registration, updates, and ownership transfers. Validation mechanisms ensure compliance with domain name and IP address standards, while keccak256 hashing guarantees tamper-proof mappings. Event emissions create an immutable and transparent audit trail for all operations. Performance testing on the Sepolia testnet demonstrated the system's efficiency. Domain registration, the most resource-intensive operation, averaged 100.320 milliseconds, while ownership transfers and email registrations recorded averages of 16.014 milliseconds and 36.550 milliseconds, respectively. Gas optimization techniques minimized storage requirements and transaction costs, ensuring scalability. By removing reliance on centralized authorities, the proposed dDNS system empowers users with full ownership of their domain assets. However, challenges such as high gas fees and scalability limitations persist. Future work could explore layer-2 scaling solutions or alternative blockchain platforms to enhance cost efficiency and adoption. This research represents a significant step toward decentralizing DNS, providing a secure and scalable alternative to traditional systems, and paving the way for broader deployment.

Keywords : *Decentralized DNS, Blockchain, Ethereum, Remix IDE, Smart Contracts, Domain Management, Decentralization.*

ACKNOWLEDGEMENT

We would like to thank our faculty and management for their professional guidance towards the completion of the mini project work. We take this opportunity to thank Dr. Ashok Shettar Pro-Chancellor, Dr. P.G Tewari Vice-Chancellor and Dr. B.S.Anami Registrar for their vision and support.

We also take this opportunity to thank Dr. Meena S. M Professor and Dean of Faculty SoCSE and Dr. Vijayalakshmi M Professor and Head, SoCSE for having provided us direction and facilitated for enhancement of skills and academic growth.

We thank our guide Jayalaxmi G N Professor and SoCSE for the constant guidance during interaction and reviews.

We extend our acknowledgment to the reviewers for critical suggestions and inputs. We also thank Project coordinator Dr. Uday Kulkarni, and reviewers for their suggestions during the course of completion. We express gratitude to our beloved parents for constant encouragement and support.

Karthik K P - 01fe22bcs183
Sushmita Math - 01fe22bcs184
Harshitaa Katwa - 01fe22bcs257
Chinmay Avaradi - 01fe22bcs306

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The Domain Name System (DNS)[1][2] is essential for translating human-readable domain names into machine-readable IP addresses[3], facilitating the functioning of the internet. However, the traditional, centralized nature of DNS presents several vulnerabilities[4], such as exposure to censorship, data breaches, and system failures. In light of these issues, this report proposes a blockchain-based, decentralized DNS (dDNS)[5] system built on the Ethereum blockchain[6], aimed at enhancing security, transparency, and user control over domain registration and management. The main objective of this research is to explore the feasibility and performance of implementing a decentralized DNS solution that leverages blockchain technology. Using Remix IDE[7] and MetaMask[8], the system employs smart contracts to securely manage domain ownership and updates, ensuring tamper-proof mappings through the keccak256 hashing algorithm. This dDNS model replaces reliance on centralized authorities, giving users full ownership and control of domain assets. The methodology adopted includes the design and implementation of the system on the Ethereum blockchain[6], where domain registration, ownership transfers, and IP address validation are carried out through smart contracts. Performance testing on the Sepolia testnet was conducted to evaluate the efficiency of the system, with a particular focus on gas optimization and scalability. The results indicate that the decentralized system performs well in terms of speed and security, though challenges such as high gas fees and scalability limits remain.

1.1 Overview Of project

This project develops a Blockchain-based Decentralized DNS (dDNS) using Remix IDE to address the security risks of traditional, centralized DNS systems. By leveraging smart contracts and blockchain's transparency and immutability, the dDNS ensures secure[9], transparent domain registration, updates, and IP address mapping. The system eliminates reliance on centralized servers, giving users full control over their domains while enhancing security against attacks like DNS spoofing[10] and DDoS[11].

1. DNS: Overview and Challenges The Domain Name System is a critical part of the internet infrastructure. It converts human-readable domain names, like www.example.com, into machine-readable IP addresses, like 192.168.1.1. This helps users to navigate the internet easily by accessing websites with simple and memorable domain names rather than complicated numerical addresses.

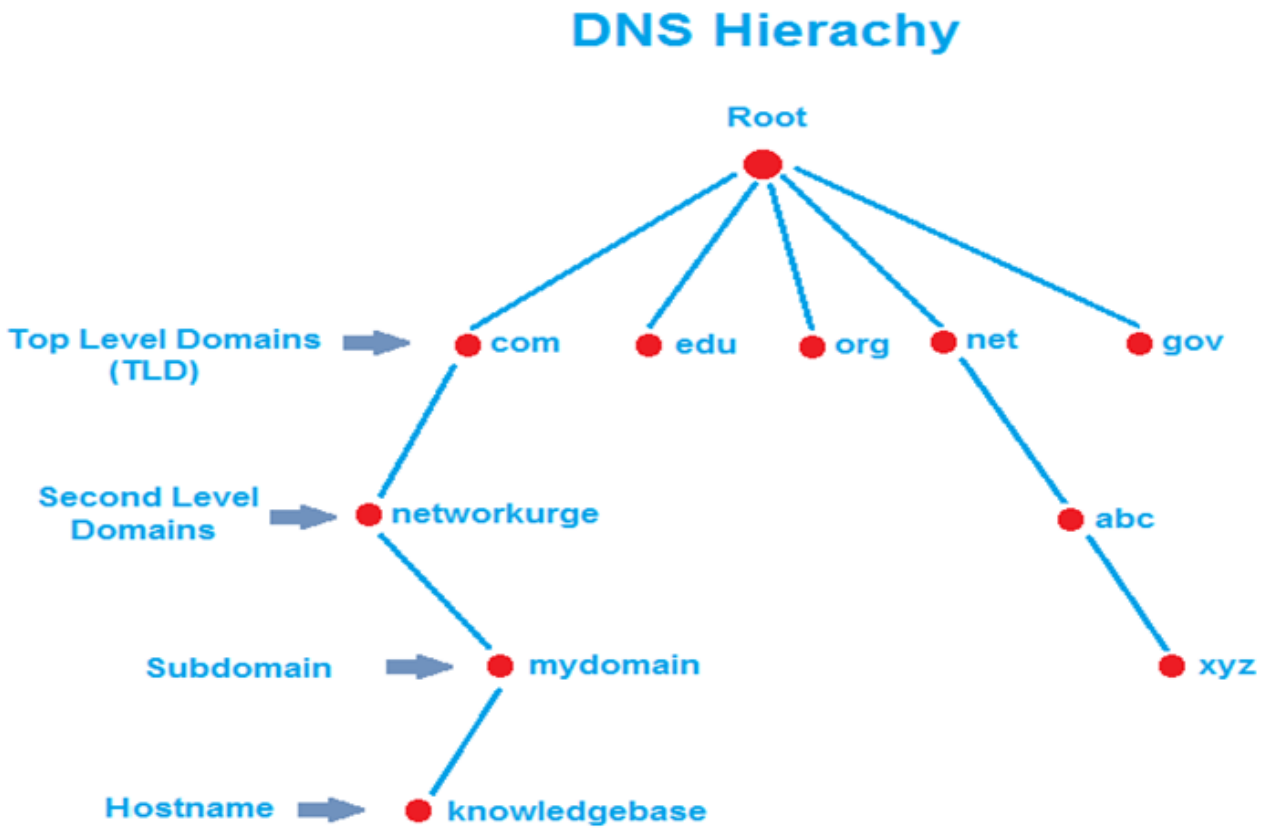

Figure 1.1: DNS Hierachy

2. Technical Limitations. DNS occupies a highly central position, but it has some technical inherent limitations which necessarily attract security and operational issues: Port

dependency: DNS relies on UDP[12] as well as TCP[13] port number 53. It makes it all vulnerable to attackers to conduct DDOS attacks, thereby really impairing the functioning of DNS.

(a) Lack of Encryption: Traditional DNS queries are not encrypted and, hence, vulnerable to eavesdropping and man-in-the-middle attacks.

(b) Latency and Bottlenecks: Centralized servers can become bottlenecks during high traffic periods, affecting response times and scalability.

DNS works like a hierarchical and distributed database that consists of several components, including:

(a) Root Server[14]: Serve the ultimate order of DNS requests and route the outputs accordingly downwards to the TLD servers.

(b) TLD servers[15]: maintain domain-specific zones such as .com, .org, and .edu.

(c) Authoritative Name Servers[16]: Maintains authoritative records for specific domains.

(d) Recursive Resolvers[17]: Serve as an intermediary for user requests to the DNS infrastructure for resolving domain requests.

3. Latest Practice and Scale: DNS underpins the global internet, processing a conservative estimate of 200 billion queries daily to serve billions of devices worldwide. It is critical to every aspect of functioning on the internet, from accessing websites to making possible cloud-based applications, email services, and IoT devices[18]. The ubiquity and scale of this technology make it an indispensable part of modern digital infrastructure.

4. Efforts to Improve DNS Innovations such as DNSSEC[19] or Domain Name System Security Extensions have dealt with these restrictions by authenticating DNS data with digital signatures, hence ensuring integrity and protection against such threats as DNS cache poisoning[20] and spoofing[10]. Although these reduce some of the attacks, they do not affect other issues such as centralization and lack of user control. Examples include DNS-over-HTTPS[21] and DNS-over-TLS[22], both adding encryption to the query itself while still relying on central infrastructures. Of the above, the critical role of DNS, current vulnerabilities and inbuilt limitations, points towards an emerging need for a more resilient and decentralized alternative. That necessity forms the bedrock for viewing blockchain-based DNS systems as offering better safety, greater transparency, and greater control for the modern Internet end user.

5. Blockchain Technology[23]: Decentralization and Security.

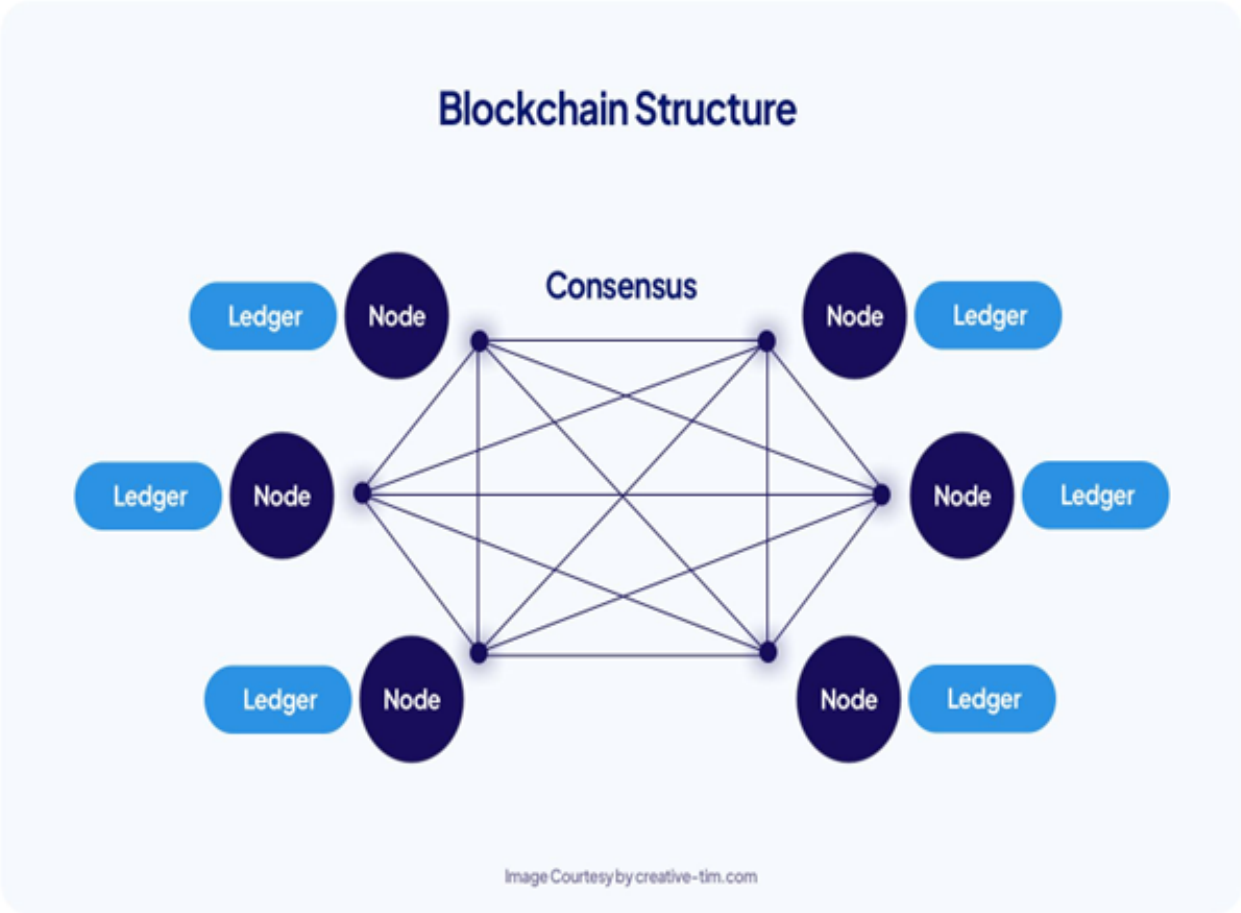

Figure 1.2: Blockchain Technology

6. How Blockchain Solves the DNS Problem The decentralized and immutable nature of blockchain makes it a perfect solution to overcome the challenges associated with the traditional DNS:

   In a nutshell, blockchain is a decentralized ledger technology that enables secure and transparent data storage across a decentralized network of nodes. This technology, from its base in cryptocurrencies like Bitcoin[24], has spread across various domains: supply chain management, voting systems, and dApps[25], which are decentralized applications

7. Main Features of Blockchain

   (a) Decentralization: Blockchain does not require a central authority because it distributes its control rights among the participants of a given network.

   (b) Immutability: Once data is recorded on a blockchain, it cannot be changed or deleted and thus provides tamper-proof records.

(c) Transparency: All the transactions and data are accessible to everyone on the blockchain, which essentially leads to accountability.

(d) Security : He cryptography helps provide data with resistance against any unauthorized changes or cyberattacks.

(e) Decentralization: No dependency on a centralized DNS authority, no more single points of failure.

(f) Tamper-proof data: guarantees that domain name records cannot be changed deleteriously .This feature has improved security by reducing DNS spoofing and cache poisoning attacks.

(g) User Control Removed: It eliminates the operational intermediary, hence giving whole control to the owners of the domain.

8. Blockchain in Practice: This would mean that the following can be applied in a decentralized DNS system: blockchain. Register domain names and updates using smart contracts. Store domain-to-IP mappings securely within the blockchain ledger. Enable transparent ownership transfers without third-party involvement.

   With blockchain, the DNS's systems will be turned more resilient, transparent, and secure, which will enable a truly reliable, censorship-resistant internet.

1.2 Motivation

The traditional DNS is based on a centralized model, which creates inherent vulnerabilities in the operation of the system to some of the most common risks, such as infrastructure collapses, cyberattacks like Distributed Denial of Service, and censorship from governments and other organizations. Centralized models create single points of failure that compromise the overall resilience and reliability of a system. Blockchain technology is quite promising in introducing decentralization to the system, which disperses it across a network of nodes, ensuring high availability, improved fault tolerance, and robust protection against attacks.

The blockchain-based DNS is transparent in nature and, therefore, has this advantage. The blockchain offers an immutable and publicly available ledger that ensures clear ownership records for domains, prevents disputes over domain ownership, and fosters

trust between stakeholders. This is way better than traditional DNS, where lack of visibility could lead to conflicts and misunderstanding.

Blockchain technology also addresses critical security vulnerabilities inherent in traditional DNS, including DNS spoofing and unauthorized modifications. Utilizing the principles of cryptography, blockchain ensures that only authorized users with the correct private keys can register or make changes to domain records. Cryptographic security eliminates the possibility of tampering and minimizes the chances of malicious activities.

Moreover, blockchain-based DNS reduces the dependency on intermediaries such as domain registrars and other centralized entities involved in the process of registering and managing domains. The reduction in dependency on middlemen reduces operational costs while enhancing efficiency, allowing for faster and more secure domain transactions.

In addition to the technical advantages, a blockchain-based DNS resists censorship, thereby promoting free expression. This is very different from the traditional system where control by a single central entity can be exerted to restrict access to certain domains by governments or organizations. Decentralization of blockchain makes it close to impossible for any singular entity to do so, thereby creating a more open, inclusive, and democratic internet, where people and communities are free to share information.

Overall, blockchain-based DNS represents a transformative shift in the way domains are managed and secured, offering a more transparent, secure, cost-effective, and censorship-resistant alternative to the traditional DNS model.

1.3 Literature Survey

1.3.1 Use Chains to Block DNS Attacks: A Trusty Blockchain-based Domain Name System

The paper "Use Chains to Block DNS Attacks: A Trusty Blockchain-based Domain Name System"[26] explores a blockchain-based approach to enhance DNS security against phishing, spoofing, and hijacking. Using Quorum's consortium blockchain and Raft consensus, it offers tamper-proof records, energy efficiency, and secure DNS resolution without major infrastructure changes. While scalable, it requires resource-intensive identity verification. This study informs our research by showcasing blockchain's potential in securing DNS systems effectively.

1.3.2 A Data Storage Method Based on Blockchain for Decentralization DNS

The paper "A Data Storage Method Based on Blockchain for Decentralization DNS"[27] presents a blockchain-based approach to enhance DNS security by addressing vulnerabilities like single points of failure and tampered data. Its advantages include fault tolerance, data integrity, and compatibility with existing DNS systems, though scalability and latency remain challenges. This study informed our project by providing key insights into decentralized DNS design and implementation, helping us build a secure and robust blockchain-powered DNS system.

1.3.3 Review of Blockchain-Based DNS Alternatives

The paper "Review of Blockchain-Based DNS Alternatives" [28]evaluates blockchain-powered DNS systems like Namecoin and Blockstack, which address traditional DNS vulnerabilities such as DDoS attacks, DNS spoofing, and centralized points of failure. These systems excel in decentralization, enhanced security, and privacy protection by eliminating protocol flaws and enabling local data storage, reducing the need for external queries. However, they face limitations like 51% attacks, scalability challenges, and the "last-mile problem," which hampers practical deployment. This review guided our project by highlighting the strengths and limitations of existing solutions, allowing us to design a more robust and scalable blockchain-based DNS model.

1.3.4 Blockchain-Based DNS

The paper "Blockchain-Based DNS: Current Solutions and Challenges to Adoption"[29] examines various blockchain-powered DNS solutions, including Namecoin, Blockstack, and Handshake, emphasizing their potential to address centralization, security, and trust issues inherent in traditional DNS. BDNS solutions provide improved decentralization, censorship resistance, and transparency, with features like tamper-proof records and resilience against cyberattacks. However, they face challenges like scalability, integration with existing DNS infrastructure, regulatory hurdles, and usability issues. This review was instrumental in identifying key design elements and challenges, enabling us to address limitations like scalability and usability in our blockchain-based DNS project.

1.3.5 B-DNS: A Secure and Efficient DNS Based on the Blockchain Technology

The paper "B-DNS: A Secure and Efficient DNS Based on the Blockchain Technology"[30] proposes a blockchain-based DNS to address vulnerabilities like cache poisoning and DDoS attacks in traditional systems. By employing a Proof-of-Stake consensus and an index for efficient querying, it enhances security and performance. Advantages include reduced attack success rates, increased attack costs, and minimized vulnerabilities, though challenges like record immutability and blockchain-specific risks remain. This study guided our project by highlighting blockchain's potential for DNS security and inspiring solutions to address its limitations.

1.3.6 DNS Service Model Based on Permissioned Blockchain

The paper "DNS Service Model Based on Permissioned Blockchain"[31] introduces a decentralized DNS framework utilizing permissioned blockchain technology. The proposed TLDChain model decentralizes top-level domain name services while enhancing security and query efficiency through data synchronization algorithms and a relational data warehouse. The study's advantages include improved data consistency and reduced query delays compared to traditional DNS models. However, the approach faces challenges like system complexity and potential data loss in warehouse synchronization. This research informed our project by emphasizing the benefits of decentralized architectures and optimization strategies for efficient domain name resolution in blockchain systems.

1.3.7 A Brief Review of DNS, Root Servers, Vulnerabilities, and Decentralization

The paper "A Brief Review of DNS, Root Servers, Vulnerabilities, and Decentralization"[32] examines the foundational role of DNS in the internet's infrastructure, highlighting its structure, root servers, and associated vulnerabilities. It discusses potential risks to DNS from cyberattacks, such as those experienced during the Colonial Pipeline ransomware attack, and emphasizes the critical need to secure DNS for safeguarding global connectivity. The study proposes research ideas for enhancing the security of DNS and mitigating its vulnerabilities. While the paper provides valuable insights, its focus on educating about DNS's current workings may lack extensive experimental data. This research has informed our project by deepening our understanding of DNS vulnerabilities and inspiring solutions for decentralization and enhanced security in DNS frameworks.

1.3.8 Improving PKI, BGP, and DNS Using Blockchain

The paper "Improving PKI, BGP, and DNS Using Blockchain: A Systematic Review"[33] explores blockchain's potential to address vulnerabilities in critical internet backbone components such as Border Gateway Protocol (BGP), Domain Name System (DNS), and Public Key Infrastructure (PKI). By decentralizing these traditionally centralized systems, blockchain enhances their security and resilience against attacks. The study highlights blockchain's ability to provide robust solutions to challenges posed by centralization while improving operational continuity. However, the implementation complexity and potential scalability issues of blockchain are acknowledged. This research has informed our project by demonstrating blockchain's transformative impact on DNS security, guiding us toward a decentralized and tamper-resistant system design.

1.3.9 The paper "DNS-BC: Fast, Reliable, and Secure Domain Name System Caching System Based on a Consortium Blockchain

The paper "DNS-BC: Fast, Reliable, and Secure Domain Name System Caching System Based on a Consortium Blockchain"[34] introduces DNS-BC, a blockchain-based DNS caching system designed to improve performance, accuracy, and security. By leveraging consortium blockchain mechanisms, DNS-BC achieves high real-time performance and 100% accuracy through an immutable credibility management system. It also employs encrypted transmission protocols to address common security and privacy concerns while mitigating traffic loads on name servers, reducing vulnerability to DoS attacks. The newly proposed DNS over KCP (DoK) protocol further enhances data transmission speed, outperforming traditional protocols like DNS over TLS (DoT) and DNS over HTTPS (DoH). This research has influenced our project by demonstrating blockchain's potential for secure and efficient DNS systems and guiding the integration of advanced encryption and performance optimization strategies.

1.4 Problem Statement

To design Blockchain-Powered Decentralized DNS for Enhanced Security

1.4.1 Problem Definition

The traditional centralized DNS system is vulnerable to multiple security risks such as DDoS attacks, DNS spoofing, and domain hijacking. These issues compromise the

integrity of domain name management and can lead to significant disruptions in internet services. Furthermore, the centralized nature of DNS leads to a lack of transparency and control for users, who are often unaware of how their data is being managed. Therefore, there is a pressing need to explore alternative solutions that provide enhanced security, transparency, and user autonomy.

1.5 Objectives and Scope of the project

1.5.1 Objectives

(a) To explore and assess the feasibility of implementing a decentralized DNS system using blockchain technology to mitigate the vulnerabilities present in traditional DNS

(b) To design a blockchain-based architecture for DNS that distributes the domain registration and resolution process, eliminating central points of failure.

(c) To create a secure smart contract framework that enables transparent and tamper-proof domain registration and ownership management on the blockchain.

(d) To analyse the performance of the decentralized DNS system in handling high traffic loads, scalability challenges, and gas optimization on the Ethereum network

1.5.2 Scope of the project

The project is going to develop a blockchain-based decentralized Domain Name System to address the issues in the traditional DNS, including security vulnerability, centralization risk, and lack of user control. The key features are: secure domain registration, updates, and ownership transfers via Ethereum smart contracts, tamper-proof record storage through keccak256 hashing, and decentralized management that would resist attacks like DNS spoofing and DDoS. The system ensures transparency through event logging, optimizes costs with gas-efficient designs, and provides robust validation for domains and IP addresses. Further scope includes scalability enhancements and broader adoption for improved efficiency and security.

Chapter 2

SOFTWARE REQUIREMENT SPECIFICATION

The SRS acts as a base document that explains the functional and non-functional requirements of the project. This chapter gives an overall view of the system specifications, ensuring that all stakeholders and developers are aligned. The SRS is the most important document that outlines the scope, constraints, and functionalities of the proposed blockchain-based DNS system.

2.1 Overview of SRS

The SRS document outlines all the requirements needed to build the blockchain-based DNS system. It is a formal agreement between the stakeholders and the development team that ensures alignment on project objectives, scope, and deliverables. The SRS is essential in reducing ambiguity, risk, and offering a structured approach to the design, development, and testing of the system.

This document includes functional and non-functional requirements for the system, which is explained by detailing its performance expectation, security measures, and usability standards. It includes comprehensive descriptions of use cases, system constraints, and dependencies to gain an overall understanding of the system's behavior. Moreover, hardware and software specifications that would be required to develop and deploy the system are described. This system, therefore, aims to enhance the security and transparency of domain name resolution by leveraging blockchain technology, reduce reliance on centralized authorities, and offer a robust and tamper-resistant solution to improve the reliability of DNS services.

2.2 Requirement Specification

The functionalities and behaviors the blockchain-based DNS system is supposed to exhibit are specified in the requirement specifications. These requirements serve as a detailed guide for the development team, listing both the system's objectives and the expected outcomes. The requirements usually fall into two categories: functional and non-functional requirements.

Functional Requirements should express the main operations, communications, and interactions in the system. They articulate certain tasks that the system has to accomplish, such as resolving domain names, records of blockchain, and other safe transactions. These serve to ensure that the systems attain their core objectives; the system provides the stipulated services to the various users. Examples of this functional requirement include secure name domain record storage and decentralized, blockchain-based management of domain name system queries.

The functional requirements are the backbone of the system development, implementation, and testing because it will ensure that the system serves its purpose and gives value to its users.

2.2.1 Functional Requirements

- The system should enable DNS registration directly on the blockchain, eliminating reliance on centralized authorities.
- Smart contracts should handle secure registration, updates, and deletion of DNS records.
- The blockchain must support resolving DNS queries using stored data.
- Only verified owners should be allowed to modify domain details, including IP addresses and canonical names.
- The system must support secure and seamless ownership transfers of domains.

2.2.2 Use Case Diagrams

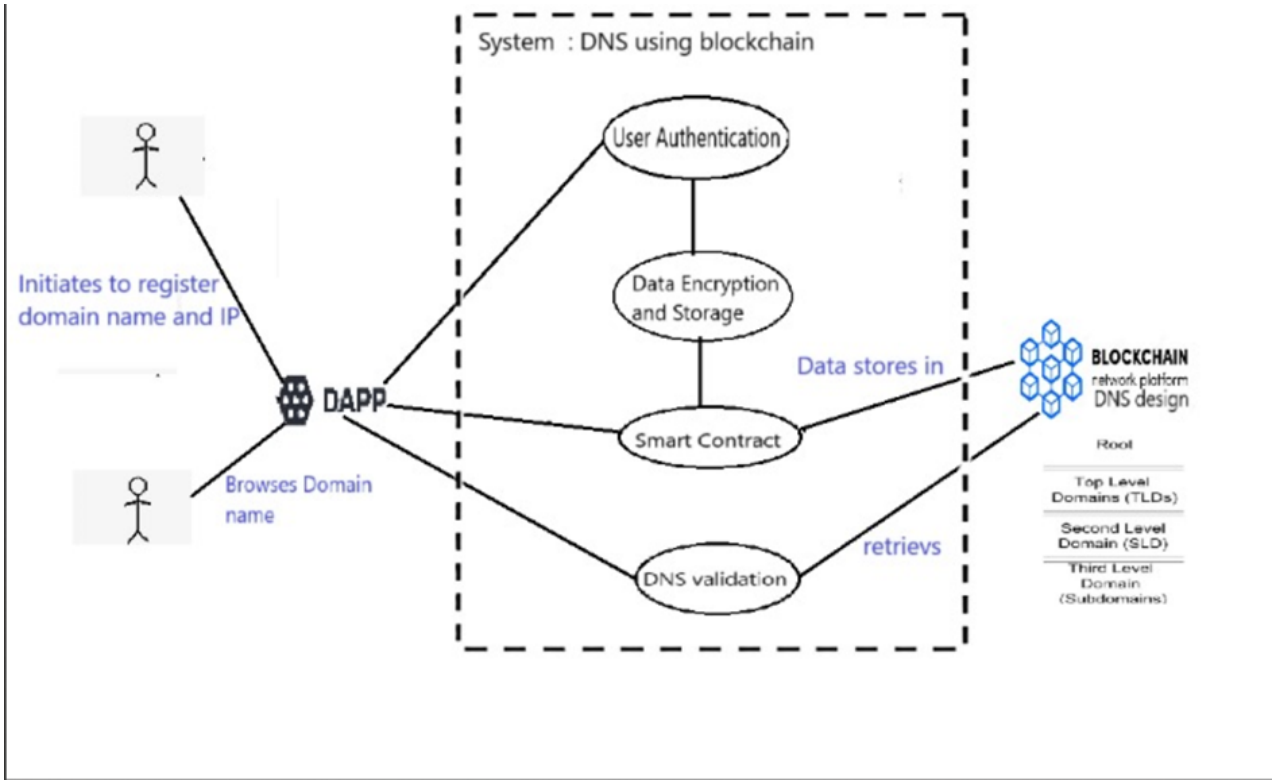

Figure 2.1: Use-case Diagram

2.2.3 Use Case 2.1 descriptions using scenarios

- Use Case Name: Domain Name Registration using Blockchain DNS
- Actors:User initiating registration Decentralized Application (DApp) Blockchain network
- Description: =A user registers a domain name and its corresponding IP address on a blockchain-based DNS system using a DApp. The system will ensure safe storage and validation through smart contracts.
- Preconditions:The user is authenticated by the system. The DApp is connected to the blockchain DNS platform.
- Flow of Events:
  (a) The user opens the DApp and initiates the registration process.
  (b) The system authenticates the user.
  (c) Data of the user (domain name and IP) is encrypted and stored in the blockchain.

(d) A smart contract validates and confirms the DNS registration.

(e) Data is safely stored in the blockchain network.

(f) Postconditions: The domain name is registered and available on the blockchain-based DNS.

(g) The user receives a confirmation of successful registration.

Exceptions:

(a) Authentication failure due to invalid credentials.

(b) Network issues during data storage or retrieval.

(c) Smart contract validation error for incorrect domain/IP format.

2.2.4 Non Functional Requirements

- Use the blockchain's immutable nature and keccak256 hashing for tamper-proof storage and validation of domain-IP combinations.
- Optimize the smart contract to minimize gas costs during storage and event-driven operations.
- The solution must support a growing number of domains and transactions without performance degradation.
- Ensure fast execution times for operations such as domain registration, query resolution, and ownership validation.

2.3 Hardware Requirements

- Processor: Quad-core CPU; may be either Intel i5/i7 or AMD Ryzen 5/7
- RAM: Minimum 8 GB (16 GB recommended for smooth multitasking)
- Storage: 256 GB SSD (512 GB or higher recommended for blockchain-related data)

2.4 Software Requirements

- Operating system: Windows 10/11, macOS, or Linux (Ubuntu 20.04+ recommended for Ethereum development).
- Development Tools: Remix IDE: Web-based IDE for writing and testing Solidity smart contracts. Visual Studio Code: Optional for the writing and management of files for smart contracts.

- Programming languages: Solidity: The language used for writing smart contracts. JavaScript/TypeScript: The front-end and integration languages with smart contracts.
- Blockchain Tools: MetaMask: Browser extension for Ethereum Blockchain interaction and signing of transactions. Ethereum Testnet (Sepolia): For testing smart contracts using test Ether.

Chapter 3

SYSTEM DESIGN

The system is decentralized, where the registration, updates, and resolution of domains are maintained through Ethereum blockchain smart contracts. Users interact with the system through a DApp or API, ensuring transparency, security, and resistance to tampering.

3.1 Low Level Design

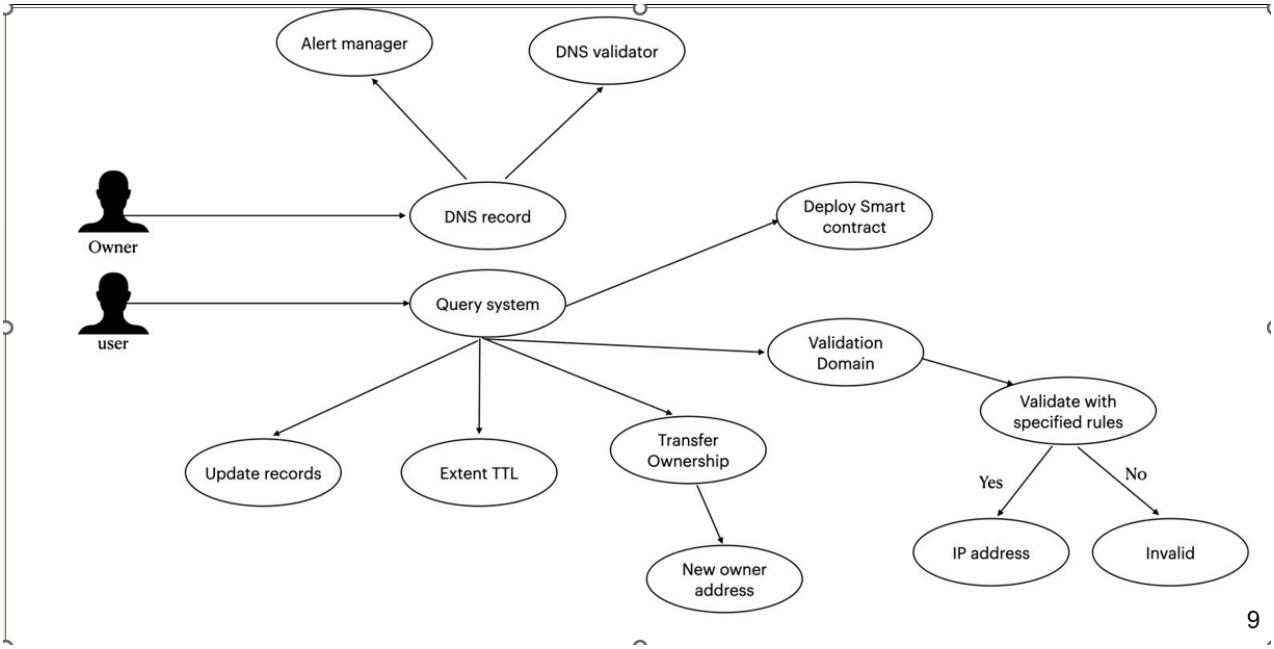

Figure 3.1: Low Level design

The figure3.1 represents a blockchain-powered decentralized DNS system that is designed to enhance security, transparency, and reliability. The system involves key components, such as the Owner, who manages DNS records by registering, updating, transferring, or deleting domains, and the User, who queries the system to validate domains and retrieve DNS information securely. The DNS records are stored on the blockchain to

ensure immutability and decentralized management. The central query system allows all interactions between the owner, user, and blockchain records. It updates records, allows for TTL extension, permits secure transfers of ownership, and validates domains. Validation involves predefined rules, such as ensuring IP addresses fall in specific ranges, like Class A or B. On successful validation, the system returns the IP address, and invalid records are flagged for prevention of unauthorized entries. Other features of the system include the Alert Manager and DNS Validator. The Alert Manager tracks important events like record update or TTL expiry. The DNS Validator ensures that records exist by cross-verifying it with blockchain data. On the Ethereum Sepolia testnet, the smart contracts deployed ensure immutability, decentralization, and transparency. They ensure tamper-proofing of the records and public verifiability. The solution addresses vulnerabilities with traditional DNS systems, such as centralization risks, DNS hijacking, and spoofing attacks. Decentralizing DNS management and embedding validation mechanisms strengthen the system to ensure stronger security, reliable ownership, and resistance to tampering or unauthorized modifications. Validation of email formats and IP ranges supports the integrity of the overall system, providing a highly robust and secure alternative against conventional DNS infrastructure.

3.2 Architecture Design

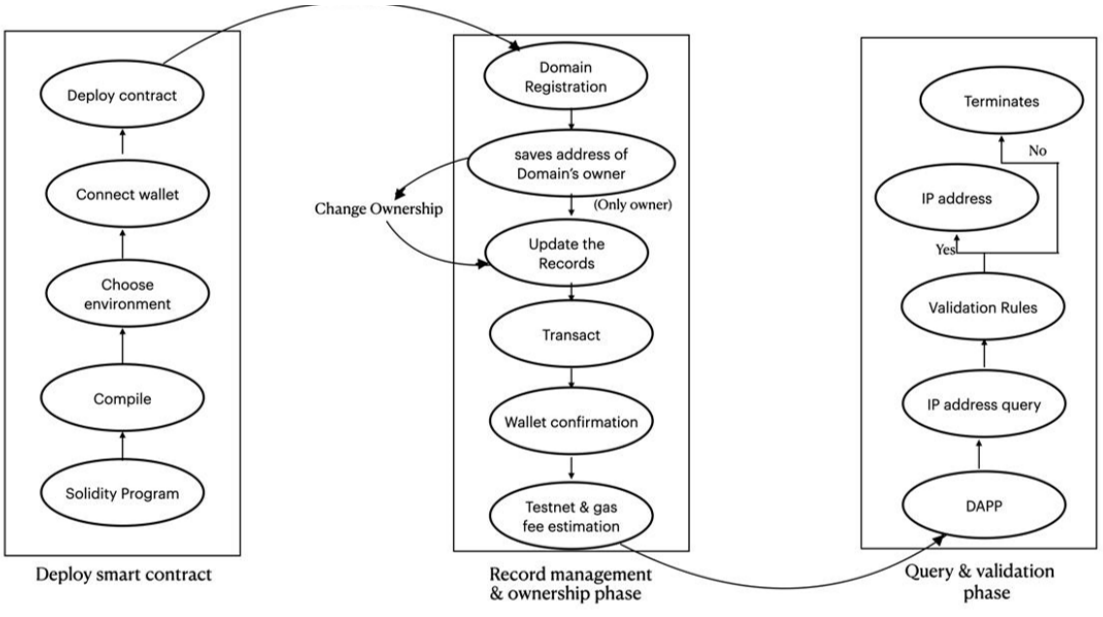

Figure 3.2: Architecture design

The figure3.2 represents the architecture design for a blockchain-powered decentralized Domain Name System (DNS), targeted to increase security. It gives an overview of the processes and components engaged in deploying and managing the system through smart contracts. Further down, an explanation will be given with respect to the problem statement:

(a) Deploy Smart Contract Phase: This phase includes writing and deploying a smart contract on the blockchain. The steps are as follows: Solidity Program: The process starts with writing a smart contract in Solidity, which is a programming language used for Ethereum-based blockchains.

   i. Compile: The Solidity code is compiled into bytecode, which the Ethereum Virtual Machine (EVM) can execute. Choose Environment: Choose an environment to deploy a smart contract. This is typically done through tools like Remix IDE, connected to a blockchain network such as Sepolia Testnet.

   ii. Connect Wallet: The user connects his wallet, for instance MetaMask, to the selected blockchain environment. It will be used for signing transactions and for paying gas fees. The compiled smart contract is deployed to the blockchain. This would ensure the contract is stored immutably upon the decentralized ledger.

(b) Record Management and Ownership Phase: This phase oversees the registration of domains, updating records, and the transfer of ownership.

   i. Domain Registration When a domain is registered, the smart contract keeps the address of the domain's owner. This ensures that the ownership is transparent and verifiable.

   ii. Saves Address of Domain's Owner: The smart contract stores the address of the domain owner safely to ensure that no unauthorized change takes place.

   iii. Update the Records: The owner can update DNS records, such as mapping a domain name to an IP address. Only the owner can do this, ensuring safety and integrity.

   iv. Transact: Transactions are created on the blockchain for updating records or transferring ownership. Each transaction incurs gas fees.

   v. Wallet Confirmation: The wallet confirms and signs transactions to ensure they are authorized by the owner

   vi. Testnet and Gas Fee Estimation: The transactions are tested on a testnet, such as Sepolia, to estimate gas fees and verify the behavior of the smart contract before it is executed on the mainnet.

(c) Query and Verification Stage: This stage allows a query for domain information and authenticating it.

i. DAPP (Decentralized Application) A front-end application referred to as DApp where users can query the smart contract to obtain domain information.

ii. IP Address Query Through the DApp, this interacts with the smart contract by fetching the IP address for a given domain name.

iii. Validation Rules: Validation rules are applied so that the IP address matches the right domain. This validates the integrity of the DNS records.

iv. IP Address: If the query results pass the validation rules, the IP address is returned. Otherwise, the query terminates.

v. Terminates: If validation fails, the process terminates without returning bad data.

This traditional DNS architecture utilizes centralized servers, thus is vulnerable to attacks like DNS spoofing, DDoS, and unauthorized access. By utilizing blockchain technology and smart contracts, this architecture makes sure that:

(a) It uses decentralization records for DNS are stored on a blockchain.

(b) Transparency: Ownership and records of changes are traceable and verifiable on the blockchain.

(c) Security: Updating records is performed only by authorized users-the owners-and ensures data integrity.

This design writes and deploys the smart contract using Remix IDE, controls transactions using MetaMask, and tests using Sepolia Testnet.

3.3 System Modelling

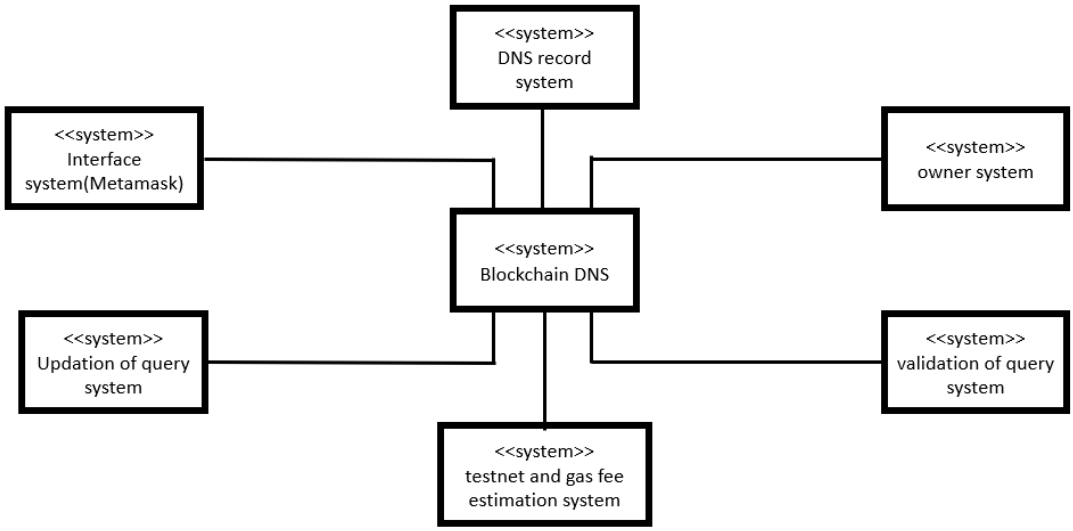

Figure 3.3: System Modelling

The following diagram depicts a Blockchain DNS System3.3 and its subsystems with mutual dependencies. The following describes the individual systems and their role within the architecture:

(a) DNS Record System: The system is responsible for registering and storing domain names in the blockchain. It enforces uniqueness of domains for owners using smart contracts and also manages operations such as querying the domains and retrieving data.

(b) Owner System: The system represents the owner's system, which governs the ownership of the domain names. Allows secure transfer of ownership and ensures that only verified users can update DNS records. Integrates with wallet systems, such as MetaMask for verification of ownership via cryptographic signatures

(c) Interface System (MetaMask): This is the interface presented to the user for engagement with the blockchain DNS. It enables users to complete actions such as domain registration, retrieving domain records, and transaction fees remittance. It is also set up to integrate with other blockchain wallets for authentication as well as transaction signing purposes.

(d) Validation of Query System :

Ensures that all incoming requests, such as domain updates and registrations, are valid. Verifies user permissions and checks the integrity of submitted data. Plays a crucial role in ensuring the security and reliability of the DNS system. Updation of Query System:

Handles the updation of existing DNS records in the blockchain. Ensures consistency and accuracy of data after any updation. Works with validation and ownership systems to ensure only authorized updations are processed.

(e) Testnet and Gas Fee Estimation System:

Provides an environment for testing DNS operations on a blockchain testnet (e.g., Sepolia). Estimates transaction costs by calculating gas fees required for various operations. Helps developers optimize their contracts and users plan their transactions.

(f) Blockchain DNS (Core System):

The central system that integrates all subsystems. Manages the interaction between the DNS records, validation, updates, and external interfaces. Ensures a seamless workflow for registration, querying, and updating of domain names in a decentralized manner.

Workflow Explanation:

(a) A user uses the Interface System (MetaMask) to perform operations such as registering or updating a domain.

(b) Validation of Query System verifies the genuineness and authorization of the request made by the user

(c) Based on the type of operation, the request is sent to either DNS Record System or Updation of Query System

(d) The Owner System validates ownership and permission before making any updations.

(e) The Testnet and Gas Fee Estimation System calculate transaction costs, offering real-time fee estimations.

(f) All interactions get processed and are stored within the Blockchain DNS, so they remain open to full transparency, are immutable, and are secure.

Chapter 4

IMPLEMENTATION

The goal of this project implementation is to create a decentralized DNS (Domain Name System) via blockchain technology that guarantees security and avoidance of risks, e.g., domain hijacking and DNS spoofing. It is based on Solidity and released on the Sepolia network making the contract reliable, thus safe - thus, only registered domains are available for it to handle. The transparency and fault-tolerant structure of Ethereum due to its decentralized nature, which does not need the intervention of a centralized server, give the users a secure system of the domain name.

4.1 Proposed Methodology

The BlockchainDNS is a decentralised Domain Name System implemented on smart contracts utilising the Ethereum ecosystem. Their objective is to generate a completely secure, immovable mechanism for the process of holding domain names related with given IP addresses together with electronic mail validation; built by the blockchain. This idea is based on blockchain in that it inherits fundamental features through decentralization and its immutable status. The Domain structure stores information for a domain, including a hash of the domain name and its associated IP address (domainHash), the owner's address (owner), the registration timestamp (timestamp), the domain's IP address (ipAddress), the expiry time of the domain (expiryTime), and the canonical name (canonicalName). The Email structure is used to hold the email address (emailAddress) and its associated IP address (ipAddress). These structures are stored using mappings, allowing for efficient retrieval of domain and email information based on the domain name or email address.

The BlockchainDNS contract is designed to emit different events for better transparency and trackability. For example, a DomainRegistered event is emitted upon registration of a new domain. On the other hand, DomainUpdated is emitted in case details of a domain are updated. Similarly, DomainTransferred and DomainDeleted events are emitted on the occurrence of a transfer of ownership of a domain or on deletion of a

domain. These events can be tracked by external systems, such as dApps or front-end applications, on domain and email registration activities.

The contract permits owners to move a domain to an updated Ethereum address. The domain moving function should only be executable by a caller who currently holds that domain and provided a valid new owner's address such that the address was not the zero address. There is also provision by the smart contract to let the holders delete their domain names prior to the expiration date. When a domain becomes deleted, the mapping removes its associated canonical name, thus no longer being accessible for data.

Security is one of the design considerations of BlockchainDNS. The contract utilizes access control mechanisms to safeguard sensitive operations. The onlyOwner modifier is implemented to ensure that only the owner of a domain can update information regarding the domain, transfer ownership, or delete the domain. Further, the notExpired modifier restricts the actions on the domain after it has expired. It ensures that no one can update or transfer the expired domains.

In brief, BlockchainDNS combines traditional domain management features with the decentralized and secure properties of blockchain technology. Using Ethereum smart contracts, the system provides a reliable and transparent solution for domain and email management, with built-in mechanisms for validation, ownership transfer, and security.

The methodology for the design and deployment of the smart contract-based domain and email registration system is based on a combination of modern blockchain tools and technologies including Remix IDE, Smart Contracts, MetaMask Wallet, and the Sepolia Testnet. The tools were chosen to be easy to use with great robustness, ensuring real-world blockchain interactions before deploying them on the sepolia testnet of Ethereum.

4.2 Activity Diagram

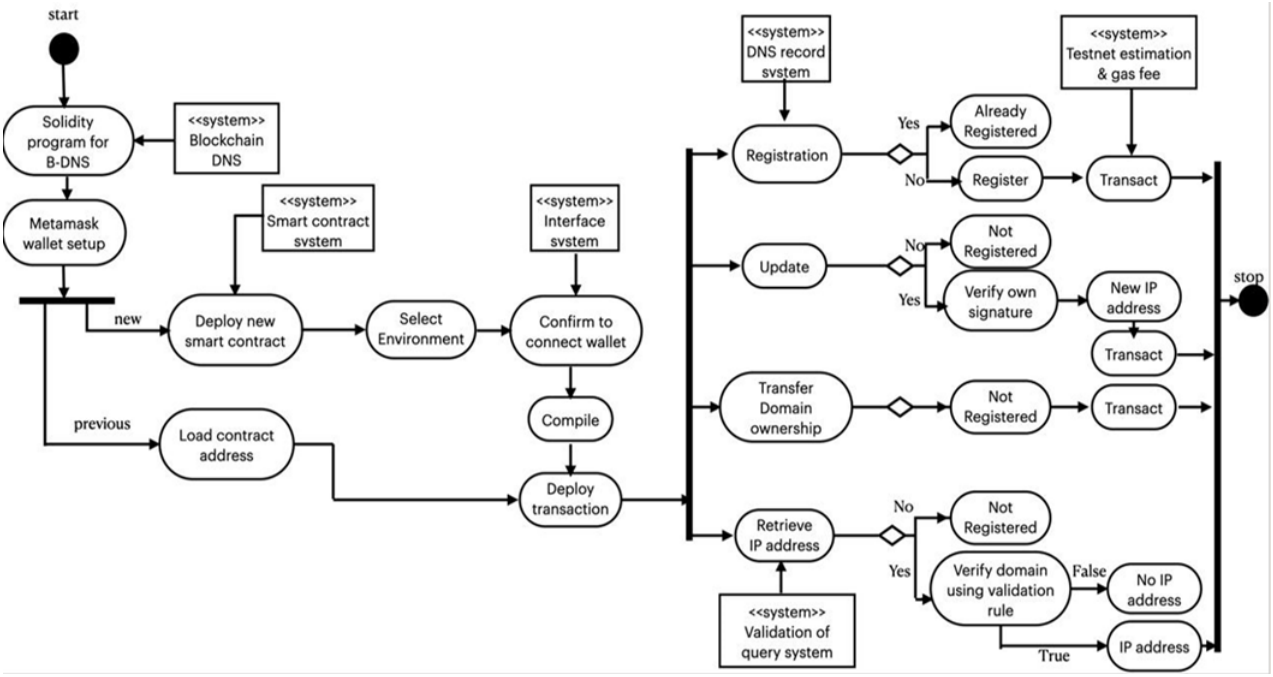

Figure 4.1: Activity diagram

The activity diagram4.1 depicts the workflow of a B-DNS system, which integrates various processes such as launching smart contracts, managing DNS records, and validating queries for a blockchain-based decentralized DNS system. Each phase described here will explain its specific role in achieving enhanced security and decentralization for managing DNS.

(a) Start Phase The process begins with the development of a smart contract, which is specifically tailored for B-DNS and coded using the Solidity programming language. This contract will determine the rules on registering domains, updating DNS records, and transferring ownership. Concurrently, the MetaMask wallet is configured to interact with the blockchain network as a tool for signing transactions and paying gas fees.

(b) Deployment Phase This phase allows users to deploy a new smart contract or load an existing one. The user either deploys a new contract by compiling the Solidity program and deploying bytecode to the blockchain network like the Sepolia testnet. Then, the appropriate environment is selected, along with MetaMask being connected to the network to safely sign transactions. If using an existing contract,

the user simply loads the contract address without needing to go through the deployment.

(c) DNS Record System Phase

The core of the B-DNS functionality is the DNS record system. During registration, the system checks if the domain is already registered. If not, the domain is registered on the blockchain, storing the owner's address and DNS records immutably. Updating DNS records, such as associating a new IP address, requires the owner to verify his identity by signing the transaction with his wallet. Furthermore, domain ownership can be transferred securely to another wallet address and the new ownership details will be updated on the blockchain.

(d) Query System Phase The query system is provided for retrieving the IP address associated with a given domain name. Upon the initiation of a query, the system checks whether the domain is registered or not. If the domain is not registered, returns the response "No IP address". For registered domains, the system validates the retrieved IP address using predefined validation rules. If the validation succeeds, the correct IP address is returned; otherwise, the query terminates without providing an invalid response. This ensures data integrity and security.

(e) Termination Phase The workflow ends after completing the required operations, such as registering, updating, or retrieving domain-related information. This phase ensures all processes conclude seamlessly, maintaining the system's decentralized and secure nature

4.3 Methods and Techniques used

The smart contract was written in Solidity language, followed by being deployed in Remix IDE and Sepolia test environment connected to Metamask for blockchain interactions. They first define the initial data structures for domains and emails. Each structure contains the important information like the owner, IP address, and expiration time.

The core features include domain registration, update, and deletion along with email validation. Each function makes sure the rules such as acceptable domain size, IP format, and ownership checks are adhered to. Modular helper functions assist in completing simpler tasks like IP validation and email parsing. Hence, scalability and user-friendliness increase due to the application of mappings and data structures that have been improved.

Testing was the main stage during the development of the blockchain-enabled DNS system which was performed on the Sepolia Ethereum test network to prove the functionality of each aspect and to ensure that it will work properly under all conditions. Because of its compatibility with Ethereum's main network and its low-cost transactions, Sepolia was the best testbed for frequent testing. The testing process was done by launching the smart contract, then through the Metamask interaction, simulate real-world scenarios using simpler data and edge cases.

4.3.1 Experimental Setup

Component/Tool	**Purpose**	**Configuration/Details**
Remix IDE	Development and testing of smart contracts	Web-based interface; used for Solidity programming and debugging smart contracts.
MetaMask Wallet	Transaction signing and gas fee payments	Integrated with Sepolia testnet for testing and mainnet for deployment.
Sepolia Testnet	Blockchain network for testing	Simulates Ethereum mainnet; uses test Ether for risk-free transactions.
Solidity Language	Writing smart contracts for blockchain DNS	Version: [Specify version used]; Ensures compatibility with Ethereum EVM.

Table 4.1: Overview of Components/Tools setup for Implementation

The table above4.1 is the summary of the main components and tools used to implement and test the Blockchain-Powered Decentralized DNS system. Each component has its part to play in the development and deployment process:

(a) Remix IDE A web-based integrated development environment (IDE) specifically designed for writing, debugging, and testing smart contracts in Solidity. It helps in smoothening the development process by features like syntax highlighting, among others, including deployment options and debugging tools.

(b) MetaMask Wallet A digital wallet used for signing blockchain transactions and managing cryptocurrencies. It is set up to use the Sepolia testnet when testing and the Ethereum mainnet when deploying. MetaMask also pays for gas fees.

(c) Sepolia Testnet This is a test blockchain network, similar to the Ethereum mainnet. It provides developers with the opportunity to test smart contracts without

risk using test Ether, thereby ensuring that the implementation works as expected before going live.

(d) Solidity Language It refers to the programming language used to code the smart contracts for the blockchain DNS system, which must support the Ethereum Virtual Machine (EVM). Its version will support all the functions and libraries, which would satisfy the project's requirements.

4.4 Implementation approach

Blockchain-based DNS comes up with plenty of security benefits as it uses the non-centralized, tamper-free, and transparent characteristics of the blockchain. Besides, these DNS systems are being reported to be centralized, which makes them prone to DNS spoofing, cache poisoning, and Distributed Denial-of-Service (DDoS) attacks. Differently, by keeping domain records on the blockchain, this decentralized DNS gets rid of the central points of failure and provides tamper-proof and immutable data storage. Every domain information, such as the IP address and canonical name, is hashed through a hashing function called keccak256, which actually makes any unauthorized modifications impossible. Ethereum's public blockchain layer is also utilized here to guarantee that the operations like domain registration and changes are clearly visible and verifiable, which helps in deterring hijacks.

DNS spoofing is securely handled by binding domains to their validated IP addresses directly on the blockchain system. The blockchain's tamper-proof mechanism brings users to authentic records rather than out-of-date or hijack-vulnerable DNS cache. The rigorous IP verification mechanism further adds to the security by strictly checking IPv4 addresses for syntactic conformity and rejecting abnormal entries from being recorded. This procedure protects the reliability of the domain-to-IP mappings, which is a core function of secure DNS.

Apart from safe mappings, the system also adopts the decentralized method of domain management thus successfully combat DDoS attacks that centralize DNS servers as their primary target. The system does not rely on a single failure-prone place to store its domain records, which are located in blockchain nodes. Counterparting, the network can manage the traffic and requests as a whole. The decentralized structure increases the protection against attacks along with hallmarking the continuity of DNS services.

The Domains expire after a fixed duration (expiryTime), calculated as block.timestamp + duration mechanism acts as an extra security layer by making the renewals be on time, thus, reducing the chances that the expired domains have been exploited for making some evildoings. Expired domains are first to be marked as inactive, which makes the unauthorized updates or misuse impossible. It creates a see-thru approach of identifying valid and actively managed domains, only the valid and actively managed domains operate. Measures like secure domain-to-IP bindings, decentralized infrastructure, thorough IP validation, and domain expiration controls enable the system to become an unassailable and invulnerable alternative to the conventional DNS architecture. It is not only overcoming the vulnerabilities such as DNS spoofing and DDoS but also builds the solid support for the next-generation communication networks, bringing trust to the blockchain.

4.5 Algorithms

4.5.1 Domain Registration Process

The system's registerDomain fuction 1 operates based on a set of rules to enable the integrity and validity of domain name registrations. Initially, a domain name has to comply with its length, which cannot be less than 4 or more than 255 characters, plus additional rules about the format such as starting and ending with an alphanumeric character and in between letters, digits, and hyphens may be used, but the hyphens must not be consecutive. The requirement of an IPv4 address being valid must be met, which is verified by a function that checks that the address consists of four octets separated by periods and the value of each octet is between 0 and 255. If the IP address is valid, a unique hashed domain along with the IP will be registered with an expiration time and a canonical name (CNAME) for the domain which will be associated with the owner. The system also checks whether the domain name has been registered or not, and if so, whether the expiration time has been set properly.

4.5.2 Algorithm for Updating Domain

In the case of the update domain function2, the procedure goes through the domain owner ,only the domain owner has the power to change the domain's IP address and canonical name this rule is specified because in order to prevent from the domain hijacking. DNS hijacking is when an attacker gets unauthorized entrance to either a DNS server or a domain registrar account and modifies the DNS records. Thus it can finally

Algorithm 1 Domain Registration Process

Require: Domain name *domain*, IP address *ipAddress*, canonical name *canonicalName*, and duration *duration*

Ensure: Registration of *domain* with associated metadata

Validate the length and format of *domain*:
if length(*domain*) < 4 **or** length(*domain*) > 255 **then**
    Throw "Domain length is invalid"
end if
if NOT (isAlphaNumeric(*domain*[0]) **and** isAlphaNumeric(*domain*[*length*(*domain*)-1])) **then**
    Throw "Domain must start and end with an alphanumeric character"
end if
for each character in *domain* excluding first and last **do**
    if character is NOT alphanumeric **and** character $\neq$ '-' **then**
        Throw "Invalid character in domain"
    end if
    if character == '-' **and** previousCharacter == '-' **then**
        Throw "Consecutive hyphens are not allowed"
    end if
end for
Check if *domain* is already registered:
if *domains*[*domain*].*owner* $\neq$ **null then**
    Throw "Domain already registered"
end if
Validate the IP address format:
if NOT validateIPv4Address(*ipAddress*) **then**
    Throw "Invalid IPv4 address format"
end if
Generate a unique hash for the domain:
$domainHash \leftarrow$ keccak256($domain + ipAddress$)
Store domain information:
$domains[domain] \leftarrow \{$
    $domainHash : domainHash,$
    $owner : CALLER,$
    $timestamp : CURRENT\_BLOCK\_TIME,$
    $ipAddress : ipAddress,$
    $expiryTime : CURRENT\_BLOCK\_TIME + duration,$
    $canonicalName : canonicalName$
$\}$
Map the canonical name to the domain:
$cnameToDomain[canonicalName] \leftarrow domain$
Emit the registration event:
Emit DomainRegistered($domain, domainHash, CALLER, canonicalName$)

Algorithm 2 Domain Update Process

Require: Domain name *domain*, new IP address *newIpAddress*, and new canonical name *newCanonicalName*

Ensure: Updates the IP address and canonical name of the *domain*

Verify if the caller is the owner of the *domain*:
if CALLER $\neq$ *domains*[*domain*].*owner* **then**
    Throw "Only the domain owner can perform this action"
end if
Validate if the *domain* has not expired:
if CURRENT_BLOCK_TIME > domains[domain].expiryTime **then**
    Throw "Domain has expired"
end if
Validate the format of *newIpAddress*:
if NOT validateIPv4Address(*newIpAddress*) **then**
    Throw "Invalid IPv4 address format"
end if
Validate the IP address class:
ipClassValidation $\leftarrow$ validateClassAorB(*newIpAddress*)
if *ipClassValidation* $\neq$ "Valid Class A IP range" **and** *ipClassValidation* $\neq$ "Valid Class B IP range" **then**
    Throw *ipClassValidation*
end if
Update the domain record:
domains[*domain*].*ipAddress* $\leftarrow$ *newIpAddress*
domains[*domain*].*canonicalName* $\leftarrow$ *newCanonicalName*
Recalculate the domain hash:
newDomainHash $\leftarrow$ keccak256(*domain* + *newIpAddress*)
domains[*domain*].*domainHash* $\leftarrow$ *newDomainHash*
Emit the update event:
Emit DomainUpdated(*domain*, *newDomainHash*, *newCanonicalName*)

direct traffic to a harmful web page or a server. Thus impact of hijacking is Legitimate sites redirection to a fraudulent one can result in a user being a victim of malicious acts that may require only a few credentials or essential information to be entered for the scam to be completed. Moreover, they may redirect the compromised traffic to engage in a variety of malicious activities or to cause service disruption. Similar to the registration step, the IP address will be examined through the same validation process to ensure it is a valid IPv4 address which is either a class A or B IP address. After the validation, the domain's details are changed, which not only involves a new domain hash but also recalculating it. The system employs these rules to ensure that unauthorized updates are impossible or with invalid data are not allowed.

4.5.3 Validation process

Algorithm 3 Domain Validation Process

Require: Input string *input* (could be a domain or canonical name)
Ensure: Validation status, associated domain, canonical name, and IP address

Determine if *input* is a registered domain or canonical name:
if $domains[input].owner \neq$ **null then**
    $domain \leftarrow input$
else if $cnameToDomain[input] \neq$ **null then**
    $domain \leftarrow cnameToDomain[input]$
else
    Return (false, "Input is neither a registered domain nor a canonical name", "", "")
end if
Retrieve the domain record:
$domainRecord \leftarrow domains[domain]$
Check if the domain has expired:
if CURRENT_BLOCK_TIME > $domainRecord.expiryTime$ **then**
    Return (false, "Domain has expired", "", "")
end if
Validate the IP address class:
$ipClassValidation \leftarrow$ validateClassAorB($domainRecord.ipAddress$)
if $ipClassValidation \neq$ "Valid Class A IP range" **and** $ipClassValidation \neq$ "Valid Class B IP range" **then**
    Return (false, $ipClassValidation$, "", "")
end if
Return validation result:
Return (true, $domain$, $domainRecord.canonicalName$, $domainRecord.ipAddress$)

when validating a domain3 first, it checks whether the domain or its canonical name exists in the database. The system ensures that the domain is active by checking if

it has not expired. If the domain has expired, then it is immediately declared invalid and the validation process4.2 is terminated. Subsequently, the system validates the IP address associated with the domain.

It checks the correct format of IPv4 for the IP address and further validates that it falls within the acceptable Class A or Class B ranges4.3. For Class A addresses, it checks if the IP address falls between 1.0.0.1 and 100.255.255.100, while for Class B, the IP range must fall between 128.0.0.1 and 170.255.255.150. The system also checks for the network as well as host portions of the IP address. For instance, in Class A, the first octet must be within the range of 1 to 100. However, any IP that does not fall within the subrange set for the first octet will be invalid, such as 100.255.255.100. In the case of Class B, the first octet must fall within the range of 128 to 170. Subrange rules apply to the second, third, and fourth octets. If the IP address does not fulfill these criteria, the domain will be marked as invalid.

The system only returns the details of the domain, including the canonical name and IP address, after ascertaining that the registration status of the domain is active and the IP address is valid, including its network and host portions. This multi-step validation ensures that domain records within the system are accurate, up-to-date, and secure, preventing unauthorized or incorrect domain and IP address mappings.

The inclusion of an IP address range validation for Class A and Class B networks will greatly be needed to ensure maintaining the security, integrity, and efficiency of the blockchain-based DNS system. Internet Assigned Numbers Authority predefined certain IP ranges to make a proper allocation and segmentation of IPs among different network sizes. By enforcing these range restrictions, the system prevents unauthorized or fraudulent IP addresses from being associated with domains, reducing the risk of malicious activities such as spoofing or man-in-the-middle attacks. Validating IP addresses also ensures that they are in the right network segments and do not conflict during domain resolution and remain compatible with the intended network infrastructure. This practice strictly follows the best practices set for network addressing to map domains to valid, legitimate addresses and ensures regulatory compliance in terms of IP address allocation. In summary, IP range validation makes the domain management system more robust against potential threats, highly secure, reliable, and organized.

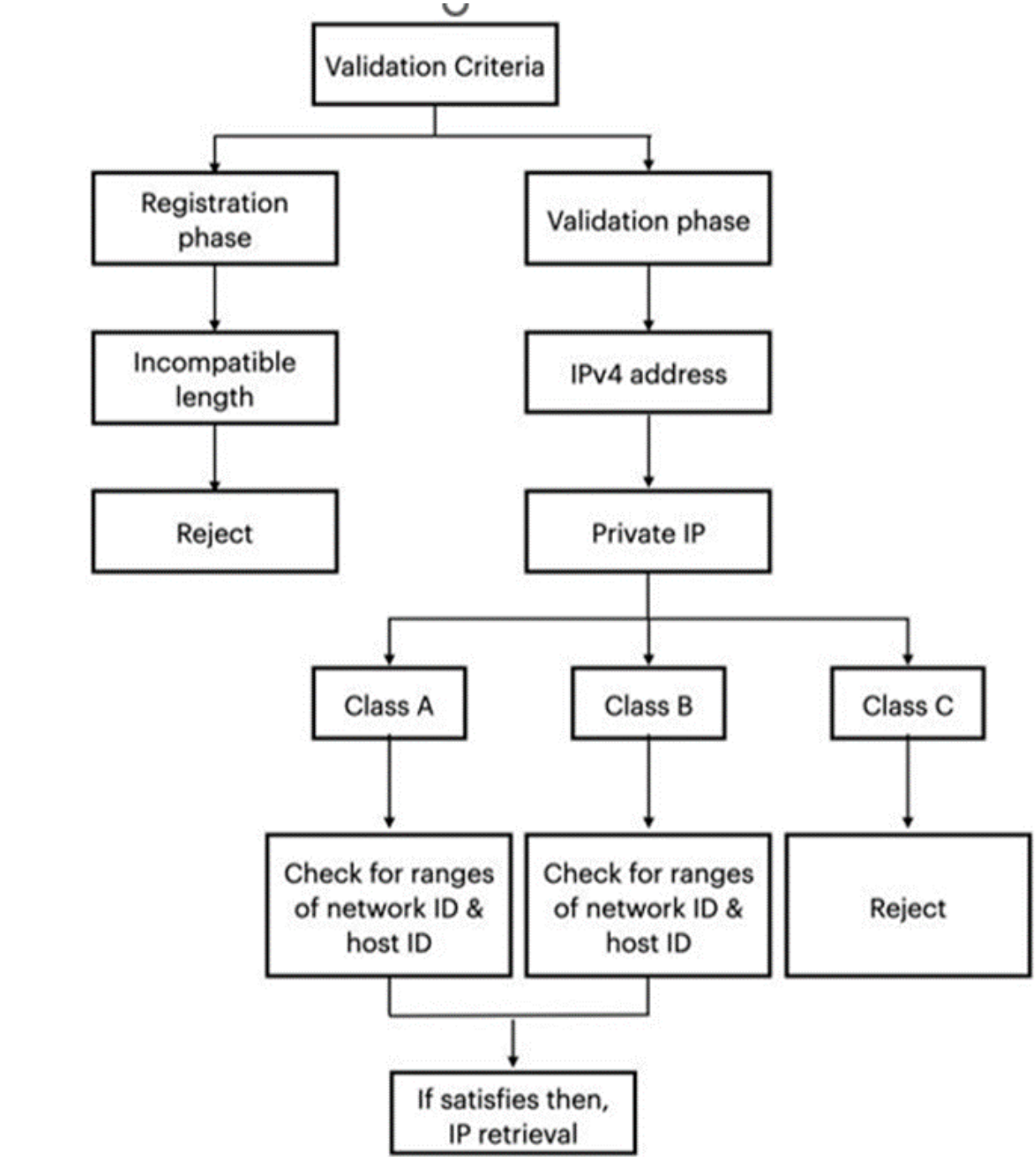

Figure 4.2: Validation criteria

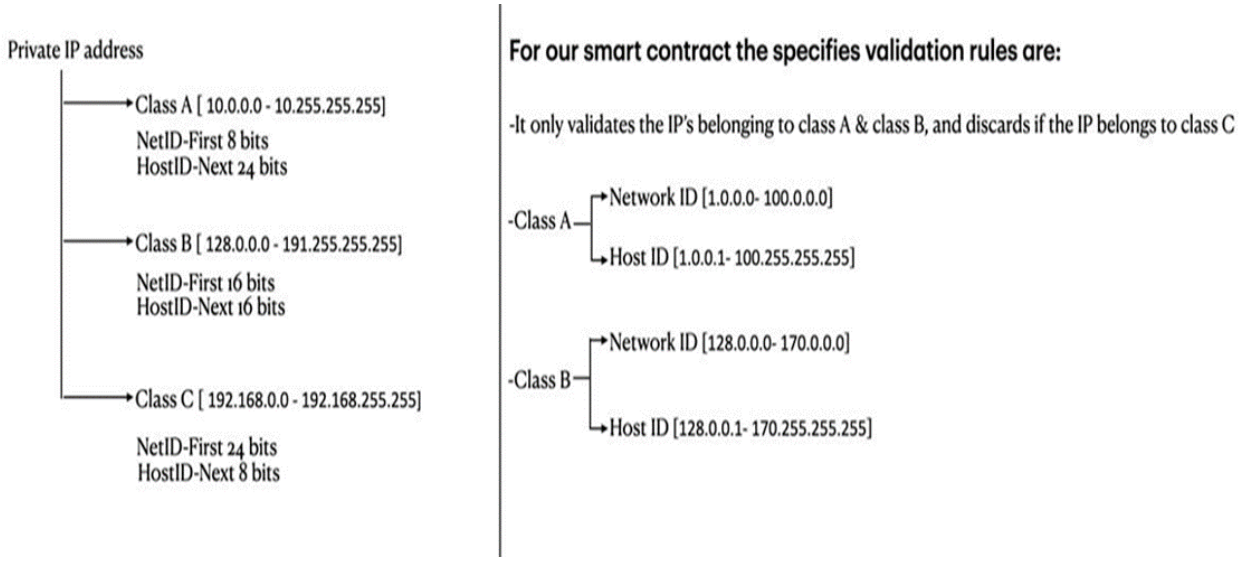

Figure 4.3: IP Addresses

4.6 Email registration and Validation

Algorithm 4 Email Registration Process

Require: Email address *email* and associated IP address *ipAddress*
Ensure: Registers the *email* with the associated *ipAddress*

Check if the *email* is already registered:
if $emails[email].emailAddress \neq$ **null then**
    Throw "Email already registered"
end if
Store the email information:
$emails[email] \leftarrow \{$
    $emailAddress : email,$
    $ipAddress : ipAddress$
$\}$
Emit the registration event:
Emit EmailRegistered($email, ipAddress$)

To register email4 the user has to provide the email address and IP when registering an email. The contract checks if the email is already registered. If not, then it stores the email and IP in the emails mapping and emits an email registered event.

Validating an email5, the system checks whether the email is registered and if it is from the domain @kletech.ac.in. If valid, it returns the email and IP; otherwise, it returns an error message. Only emails from the @kletech.ac.in domain are accepted.

Algorithm 5 Email Validation Process

Require: Email address *email*
Ensure: Validation status, *emailAddress*, and associated *ipAddress*
Check if the *email* is registered:
if $emails[email].emailAddress = \textbf{null}$ **then**
    Return (false, "Email not registered", "")
end if
Validate if the email belongs to the @*kletech.ac.in* domain:
if NOT isValidKletechEmail(*email*) **then**
    Return (false, "Email domain is not valid. Only @kletech.ac.in is allowed", "")
end if
Retrieve the email information:
$emailRecord \leftarrow emails[email]$
Return validation success:
Return (true, *emailRecord.emailAddress*, *emailRecord.ipAddress*)

Chapter 5

TESTING

Blockchain-based DNS Smart Contract Test Plan: Test that the smart contract BlockchainDNS runs successfully and securely on the Sepolia testnet with MetaMask handling the signing and management of transactions.

5.1 Test Plan

Test the following on the contract:

(a) Functionality:Domain and email registration, update, validation, transfer, and deletion.

(b) Security:Ownership verification, IP validation, and domain expiration.

(c) Performance :Handling long domain names and complex operations.

(d) Integration:Interacts correctly with MetaMask and the Sepolia testnet.

5.2 Types of Testing

(a) Unit Testing:Tests individual functions of the smart contract.

Example: Testing the functions registerDomain, validateIPv4Address, and transferOwnership individually.

(b) Integration Testing Tests interactions between functions and mappings in the contract.

Example: Registering a domain and verifying it using validateDomain.

(c) System Testing Test the entire smart contract as one unit on the Sepolia testnet.

Example: Registering a domain, transferring ownership, and verifying the domain.

(d) Acceptance Testing Verifies the contract meets end-user requirements.

Example: A user successfully registers a domain and updates the IP address via MetaMask.

(e) Security Testing Access control, prevents unauthorized actions, and validates inputs.

Example: Only the owner can delete a domain. Performance Testing Test the behavior of the contract under heavy load.

Example: Registering multiple domains and emails in quick succession.

5.3 Acceptance test plan and test cases

(a) Test Case 1: Domain Registration

i. Description: Test registering a new domain.

ii. Steps:

A. Deploy the contract on Sepolia testnet.

B. Call registerDomain with a valid domain name, IP address, canonical name, and duration.

C. Confirm the transaction in MetaMask.

iii. Expected Result:

A. The domain is registered.

B. DomainRegistered event is emitted.

(b) Test Case 2: Domain Validation

i. Description: Test validating a registered domain.

ii. Steps:

A. Call validateDomain with the domain name.

B. Confirm the transaction in MetaMask.

iii. Expected Result: The function returns true with the domain details if valid.

(c) Test Case 3: Domain Transfer

i. Description: Test transferring domain ownership.

ii. Steps:

A. Call transferOwnership with the domain name and a new owner's address.

B. Confirm the transaction in MetaMask.

iii. Expected Result:

A. Ownership of the domain is transferred.

B. DomainTransferred event is emitted.

(d) Test Case 4: Email Registration

i. Description: Test registering an email address.

ii. Steps:

A. Call registerEmail with an email and IP address.

B. Confirm the transaction in MetaMask.

iii. Expected Result:

A. The email is registered.

B. EmailRegistered event is emitted.

5.4 Unit test plan and teat cases

(a) Test Case 1: Validate IPv4 Address

i. Description: Test the IPv4 address validation logic.

ii. Inputs:

A. Valid IP addresses (e.g., "192.168.1.1", "255.255.255.0").

B. Invalid IP addresses (e.g., "999.999.999.999", "abcd.efg.hij.klm").

iii. Expected Result:

A. Returns true for valid IP addresses.

B. Returns false for invalid IP addresses.

(b) Test Case 2: isAlphaNumeric

i. Description: Test alphanumeric character validation.

ii. Inputs:

A. Characters like 'a', '1', '-'.

B. Special characters like '@', '', ' ' (space).

iii. Expected Result:

A. Returns true for alphanumeric characters (e.g., 'a', '1').

B. Returns false for non-alphanumeric characters (e.g., '-', '@').

(c) Test Case 3: Validate Class A or Class B

i. Description: Test Class A and Class B IP range validation.

ii. Inputs:

A. Class A IPs: "1.0.0.1", "100.255.255.100".

B. Class B IPs: "128.0.0.1", "170.255.255.150".

C. Non-Class A/B IPs: "200.0.0.1".

iii. Expected Result:

A. Returns "Valid Class A IP range" for Class A IPs.

B. Returns "Valid Class B IP range" for Class B IPs.

C. Returns "Invalid IP class (Not Class A or Class B)" for other IPs.

Test Case ID	**Function**	**Description**	**Inputs**	**Expected Result**
UTC-01	validateIPv4Address	Test IP validation logic	192.168.1.1, 256.256.1.1	Returns true for valid IPs, false for invalid ones.
UTC-02	generateDomainHash	Test domain hash generation	"example.com", "1.1.1.1"	Returns a consistent hash for the same input.
UTC-03	isAlphaNumeric	Test character validation	'a', '1', '-', '@'	Returns true for alphanumeric characters, false for invalid ones.
UTC-04	validateClassAorB	Test Class A/B IP validation	1.0.0.1, 200.0.0.1	Returns valid/invalid class range information.

Table 5.1: Unit Test Cases

Chapter 6

RESULTS DISCUSSIONS

The implementation of the Blockchain-Powered Decentralized DNS for Enhanced Security was successfully carried out using the Remix IDE and deployed on the Ethereum blockchain with MetaMask integration. The smart contract enabled secure registration and management of domain names, with validation mechanisms ensuring adherence to industry-standard rules for format and length. IP address validation was implemented for Class A and Class B ranges, and any invalid or out-of-range addresses were appropriately rejected, enhancing reliability. Ownership verification was enforced for modifications to domains, such as updating IP addresses and canonical names, while ownership transfers were seamlessly supported, providing decentralized control of domain assets.

Additionally, the system facilitated the registration of email addresses linked to IP addresses, with a unique feature enforcing the use of organizational email domains like @kletech.ac.in. Validation checks ensured emails met this requirement and confirmed their registration status. Security was prioritized through the use of ownership and expiration checks, ensuring that only authorized users could manage domains or emails and that expired domains were automatically invalidated. The use of keccak256 hashing for domain-IP combinations added an extra layer of security by generating tamper-proof unique identifiers.

Transparency was achieved through event emissions for actions such as registration, updates, transfers, and deletions, providing an immutable audit trail on the blockchain. Utility functions like domain splitting and IP validation were efficiently implemented, with rigorous testing confirming the system's ability to handle both valid and invalid inputs robustly. The decentralized nature of the solution eliminated reliance on centralized entities, granting users full ownership and control of their resources. Furthermore, the contract was optimized for gas efficiency through minimal storage requirements and event-driven operations. Overall, the implementation successfully demonstrated a secure and transparent blockchain-based DNS and email management system, aligning with decentralized principles while offering significant advantages in security and scalability.

(a) Performance of remix ide platform

The Remix IDE platform is very efficient and user-friendly in performance, which makes it one of the most widely used tools for developing and testing Ethereum smart contracts. Fast contract compilation is provided by using the Solidity compiler (solc) that gives real-time feedback. It is very fast for small to medium-sized contracts, but if the contracts are larger, it takes longer time for compilation. When interacting with contracts on test networks like Rinkeby ,sepolia, Goerli, Remix provides near-instant execution, allowing developers to test and deploy smart contracts with minimal delay. However, the execution speed may vary depending on network conditions and the complexity of the contract.

Remix also provides accurate gas usage estimates during contract deployment and interaction, helping developers optimize their code. These estimates are usually reliable but actual gas usage may vary when deployed on live networks as network congestion can be quite different. The simulation of transactions on the platform also provides fast feedback, allowing developers to quickly see the results of their transactions. Another interesting feature is the built-in debugger, which provides detailed information about the behavior and performance of contracts. Overall, Remix IDE does provide efficient, responsive performance for Ethereum contract development, which may vary with more complex contracts or heavier network traffic.

6.1 Mathematical Formulation

1. **Use of Gas Computation:**

$$\text{Gas\_Used} = \frac{\text{Transaction\_Fee}}{\text{Gas\_Price}}$$

Where:

- Transaction_Fee total cost for the transaction (in ETH).
- Gas_Price price per unit of gas (in ETH).

2. **Estimation of Execution Time:**

$$\text{Execution\_Time} = \frac{\text{Gas\_Used}}{\text{Gas\_Processing\_Rate}}$$

Where:

- Gas_Used: Amount of gas consumed by the transaction.

- Gas_Processing_Rate: The rate at which blockchain processes gas. Usually, this is set at about $2,000,000$ gas/second for testnets like Sepolia.

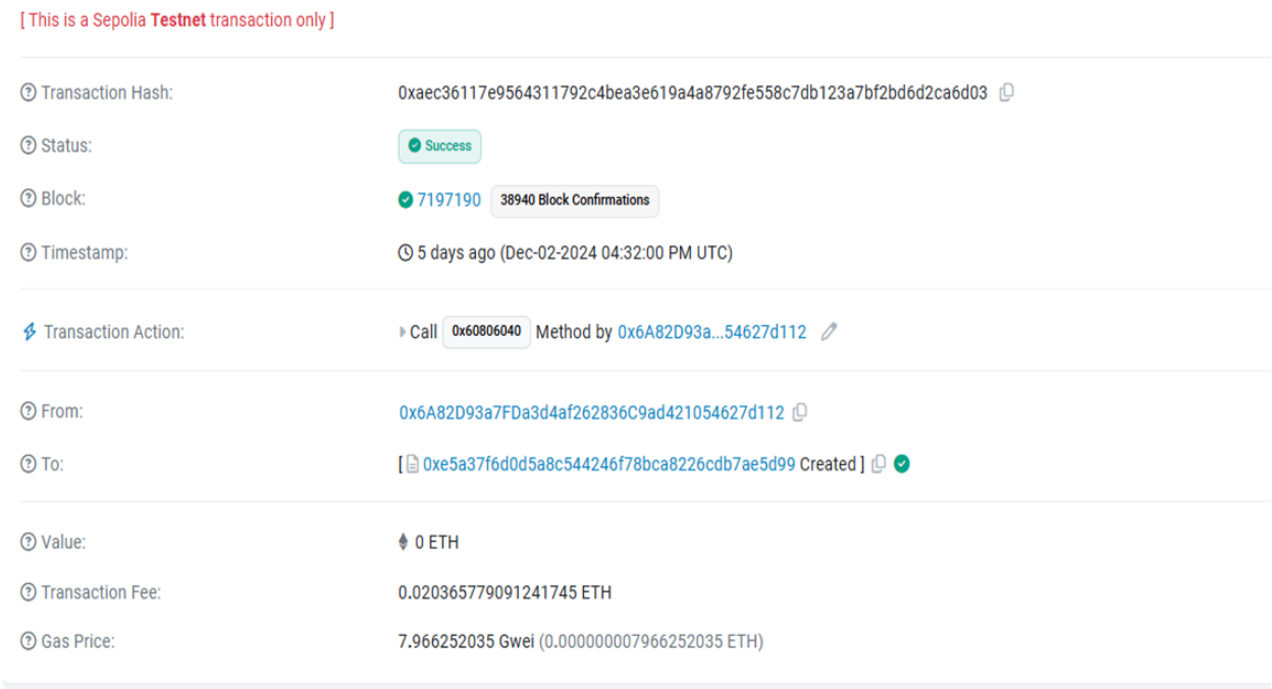

Figure 6.1: Simulation of B-Dns Smart contract in Sepolia Testnet

6.2 Evaluation of time for functions

Figure 6.2is the simulation result of a variety of smart contract functionality on Sepolia testnet, in Remix IDE with MetaMask. Tasks are analyzed; they include: Register Domain, Register Email, Transfer Ownership, and Update Domain. Each shows a range of execution time due to the complexities and the kind of interactions done with the blockchain.

(a) Register Domain: This one took the maximum time to run with an average of 100.320 milliseconds. Its length is perhaps caused by the fact that this domain involves checking whether the given domain name can be available, update the DNS, and then writing it in blockchain, which takes more gas and computations as such compared to the most common ones.

(b) Transfer Ownership: The transfer ownership functionality took the least amount of time to execute, averaging at 16.014 milliseconds. This is because, in most cases,

transfer of ownership only involves changing metadata-the owner of the domain-which is a lightweight operation with low computational overhead.

(c) Email Registration: It took an average of 36.550 milliseconds to register an email. The process is a mid-range execution time since it has to associate the email with an IP address, which does involve some level of computation but not as complicated as that involved in registering a domain.

(d) Updating a domain : Its averaged 37.030 milliseconds, which is moderately complex. This is to update associated data of the domain including its IP address and canonical name, which will be a little more computational than updating an email but less than updating a domain.

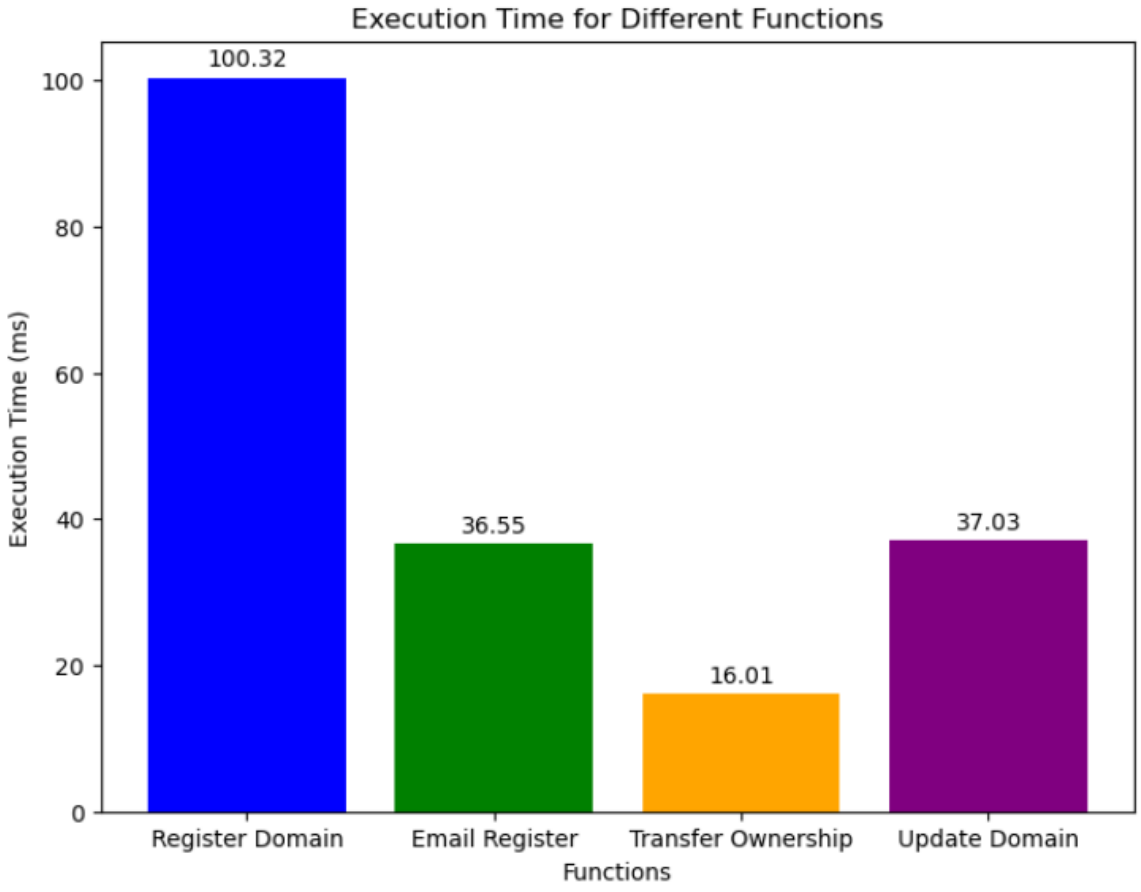

Figure 6.2: Execution Time for Function to deploy in testnet

6.3 comparison of execution time for single query Vs Multiple queries

The execution times in the figure6.3 are the times taken for the deployment of smart contracts on the Sepolia testnet for a specific functionality such as the registration of a domain or email, using Remix IDE and MetaMask wallet. Here's an explanation of what these times mean in this context:

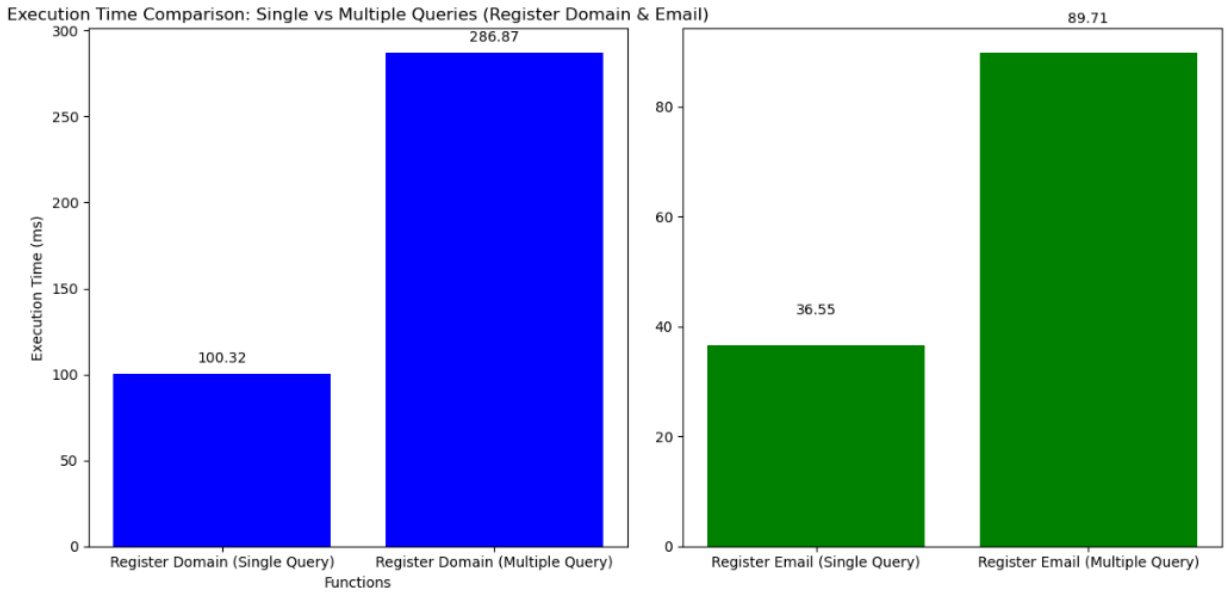

Figure 6.3: Single Query vs Multiple Queries

(a) Single Query:

Time to deploy the smart contract and execute one single transaction for the functionality e.g., registering one domain or email on Sepolia testnet. Includes in this time The compiling of the contract within the Remix IDE Its deployment on the blockchain with the MetaMask Transaction confirmation at Sepolia testnet.

(b) Multiple Queries (3 Queries at a Time):

Time taken to deploy the smart contract and execute three simultaneous transactions for the functionality. The increase in execution time reflects the added overhead of processing multiple transactions concurrently, such as: Blockchain network latency. Increased gas consumption due to more intensive operations. Verification and confirmation of multiple transactions by the network nodes.

6.4 Comparision Between Traditional DNS and Blockchain Based DNS

Aspect	Traditional DNS	Blockchain-Based DNS
Ownership Control	Only registrars and intermediaries have control.	Full ownership through smart contracts.
Validation Methods	Semi-automated with a high chance of human error.	Fully automated with strict adherence to rules.
Security Against Attacks	Prone to DNS spoofing and DDoS attacks.	Decentralized and cryptographically secure.
Transparency	Domain changes not clearly visible.	Records are immutable and auditable.
Dependency on Trust	Highly dependent on the registrars.	Smart contracts ensure trustless operation.

Table 6.1: Comparison between Traditional DNS and Blockchain-Based DNS

6.5 Dependency on Gas Consumsuptiom

The execution time and fee of transactions depends on how complex the contract is to be executed with respect to the number of queries going to be queried. Multiple queries consume more gas and time because the blockchain has to process each transaction independently while maintaining consistency and security.

Chapter 7

CONCLUSIONS AND FUTURE SCOPE

The realization of a blockchain-based distributed DNS and email management platform addresses the limitations of traditional DNS with the use of Ethereum for added security and transparency. During the project, it could be seen that the safe registration and management of the domain name can be done using a smart contract deployed on the Ethereum blockchain integrated with MetaMask. Major functionality features consisted of domain/IP validation, ownership verification processes, and registration/assignment procedures of organizational emails with a certain IP address. For strong enforceability in terms of adhering to decentralized principles like using keccak256 hash for tamper-proof storages while utilizing event emissions for achieving clear transparency, the system did really give users full ownership along with full control over their resource usage. Decentralization in fact helped in achieving resource autonomy away from central authoritative dependence. Performance optimization was achieved through the implementation of efficient gas usage and transaction management. The project, in general, demonstrated the feasibility of blockchain technology in developing scalable, secure, and transparent systems.

Future Scope: Though the current implementation satisfies its requirements, there are many improvements that can be made to the system. Future work could focus on cross-chain compatibility to support multiple blockchain networks, enhancing interoperability. Scalability can be improved by using Layer 2 solutions like rollups or sidechains to reduce gas costs and handle more transactions efficiently. Security can be further enhanced by incorporating machine learning models to detect and prevent fraudulent activities based on domain/IP behavior. Additionally, the system could be expanded for enterprise use, including support for private blockchains, which would unlock new use cases and increase adoption in industries requiring secure and transparent solutions.

Appendix

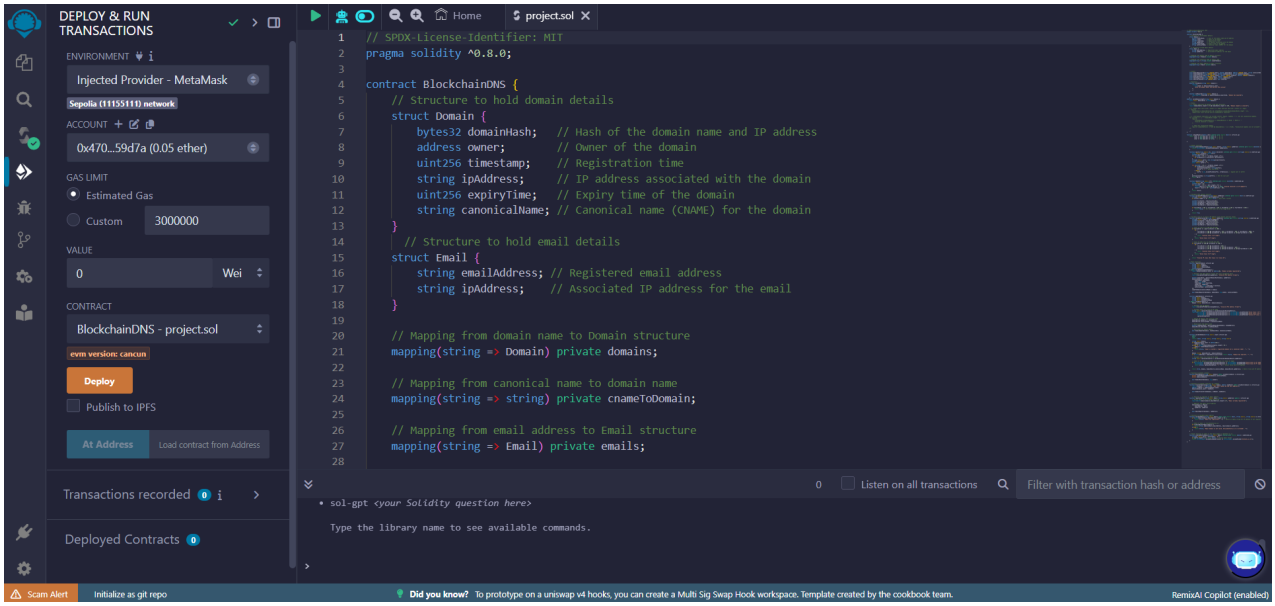

Figure 7.1: Remix IDE Platform

(a) Remix 7.1is a powerful, web-based, Integrated Development Environment designed explicitly for building Ethereum smart contracts. Supportively, it includes Solidity which is the programming language intended for writing Ethereum smart contracts. It has a feature that allows developers to interactively write, test and deploy contracts in a graphical user interface. It integrates built-in Solidity compilers, real-time debugging, and a script environment into one to interact with the Ethereum Virtual Machine (EVM). This contract for the registration of domains and emails and related functions like domain verification and IP checking was designed and debugged using Remix IDE. It made it quite easy to simulate the behavior of the contracts and debug the logic before deploying.

Figure 7.2: sepolia Testnet

(b) Smart Contracts: The smart contract is the backbone of the system, defining the logic for domain and email registration, as well as domain validation through IP address checks and email domain verification. Smart contracts are self-executing code deployed on the blockchain, meaning once they are deployed, their operations are automatically triggered by events, without the need for intermediaries. This contract in the project was written in Solidity and included features such as transfer of domain ownership, registration of emails, and verification of domains and emails. The registration process in this system is decentralized, transparent, and tamper-proof through the use of Ethereum smart contracts.

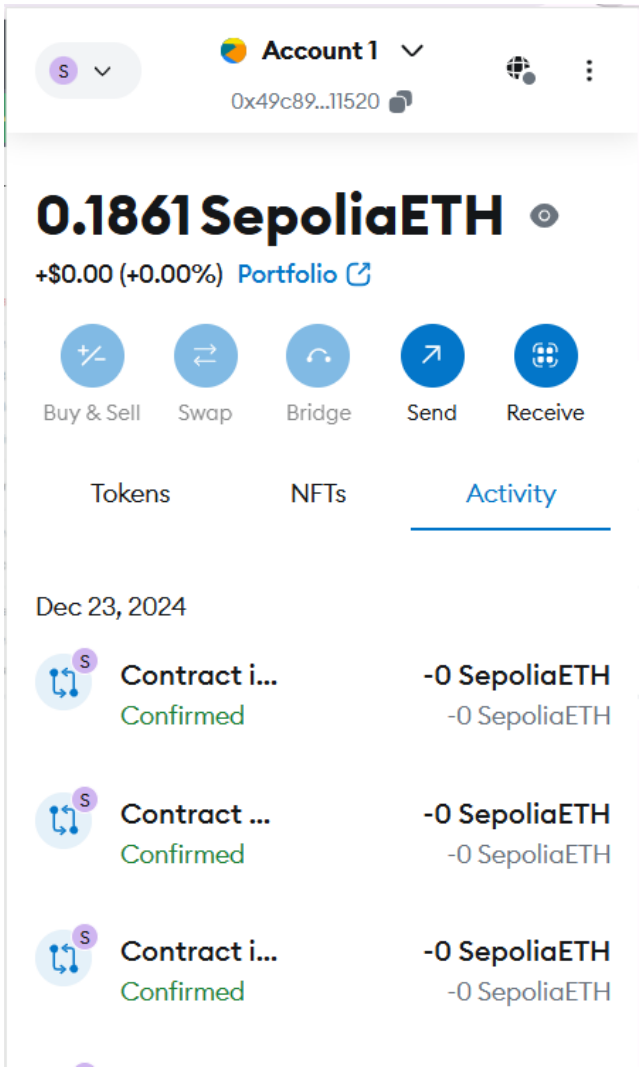

Figure 7.3: Metamask Wallet

(c) MetaMask 7.3 is a popular browser extension and wallet that enables users to interact with the Ethereum blockchain. It acts as a bridge between the user's browser and the Ethereum network, allowing users to manage their Ethereum accounts, sign transactions, and interact with dApps. In this project, MetaMask was used to deploy the smart contract on Sepolia testnet, interact with the contract for domain and email registration, and do all the necessary transactions such as paying gas fees or transferring ownership of domains. MetaMask allows interaction in the blockchain without having to run an Ethereum node.

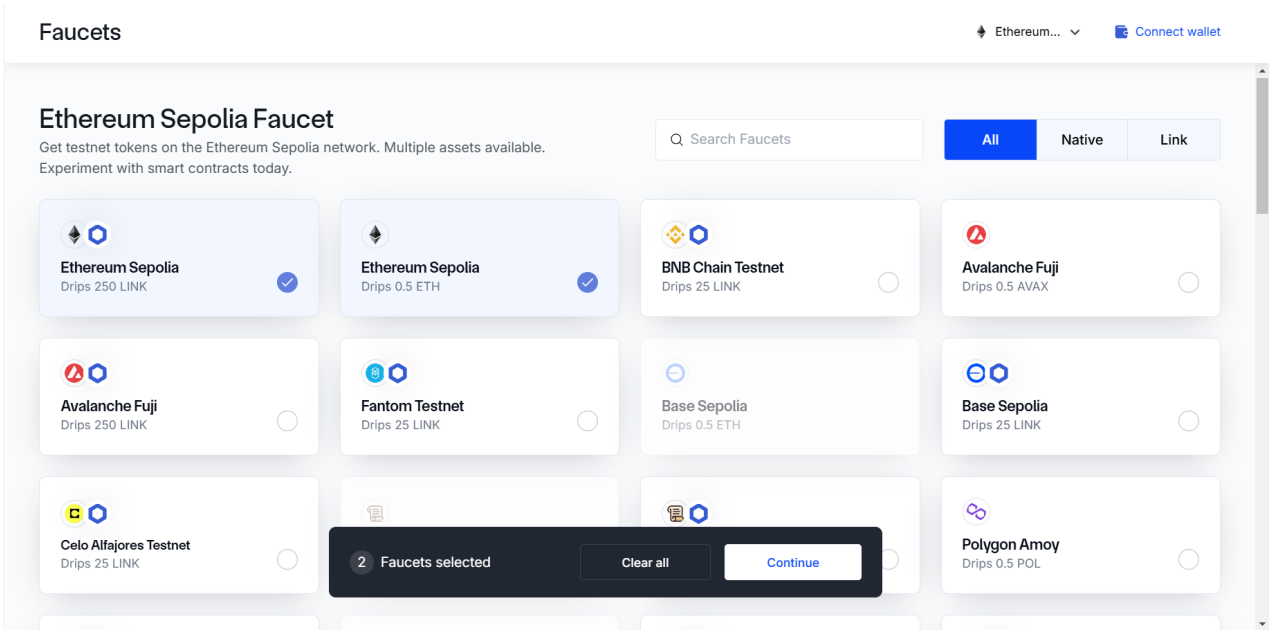

Figure 7.4: How to get sepolia Faucet

(d) Sepolia Testnet: The Sepolia testnet is a simulated environment version of the Ethereum mainnet that will help to ensure real-time testing blockchain activities without consuming any Ether in the process. Using this testnet will assist in testing the contract with regards to functionality since its execution will be just about identical to the one performed in the Ethereum mainnet. This allows for the safe and cost-effective testing of a contract in testnet before deploying it to mainnet, where real money transactions are involved. Sepolia is the same as mainnet in terms of network conditions but uses test Ether7.4 to perform transactions, which does not have any real value. This means that issues related to the smart contract would be identified and corrected without the risk of losing any real assets. All these combined gave way to a smooth development cycle. Remix IDE helped in developing and testing the contract, and MetaMask made it relatively easy to sign transactions and interact with the contract by offering a user-friendly interface, and Sepolia let me safely test at zero cost before deploying on Ethereum's mainnet. This methodology provided a secure, fully functional contract ready for real-world applications, offering a decentralized yet efficient solution for domain and email registration.

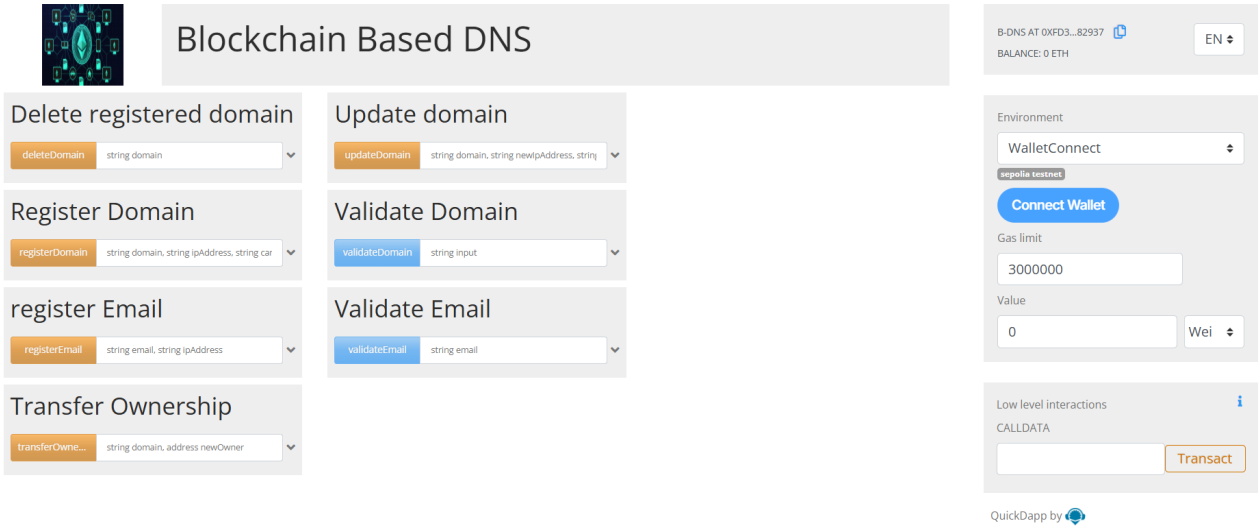

Figure 7.5: DAPP using Remix IDE

(e) DAPP

This is a blockchain-based Decentralized Application (DApp) that uses Remix IDE for development and was deployed on the Sepolia testnet. It has provided a user-friendly interface through which multiple DNS-related operations are executed, such as securely and efficiently managing domain and email records on the blockchain. The DApp enables users to create new domains and email addresses, update domains' information such as IP addresses, and metadata, and delete existing registered domains from the blockchain records. The DApp will also offer validation functions to authenticate the existence or authenticity of domains and email addresses. The DApp also has the functionality to transfer ownership of a domain to another blockchain address, making it easy for assets to be handed over.

The application integrates with MetaMask or similar wallet providers to allow users to authenticate and sign transactions directly. The environment selector allows users to select the Sepolia testnet for deployment and interactions. More advanced features include setting a gas limit for transactions, setting Ether values, and low-level calldata input for custom contract interactions. The DApp leverages blockchain's decentralized and immutable nature to provide a secure, transparent, and efficient way of DNS management without a centralized authority, thus being a system of trustless operation.

REFERENCES

[1] Thomas Callahan, Mark Allman, and Michael Rabinovich. On modern dns behavior and properties. *ACM SIGCOMM Computer Communication Review*, 43:7–15, 07 2013.

[2] Jeffrey Pang, James Hendricks, Aditya Akella, Roberto De Prisco, Bruce Maggs, and Srinivasan Seshan. Availability, usage, and deployment characteristics of the domain name system. pages 1–14, 10 2004.

[3] Patrick Lundevall-Unger and Tommy Tranvik. Ip addresses - just a number? *I. J. Law and Information Technology*, 19:53–73, 03 2011.

[4] Georgios Kambourakis, Tassos Moschos, Dimitris Geneiatakis, and Stefanos Gritzalis. Detecting dns amplification attacks. pages 185–196, 10 2007.

[5] Guang Yang. Development and application of a decentralized domain name service, 12 2024.

[6] Satpal Kushwaha, Sandeep Joshi, Dilbag Singh, Manjit Kaur, and Heung-No Lee. Systematic review of security vulnerabilities in ethereum blockchain smart contract. *IEEE Access*, PP:1–1, 01 2022.

[7] Dharm Vashisth, Parth Khandelwal, Rahul Johari, and Varnika Gaur. Blockchain technology based smart contract agreement on remix ide. 08 2021.

[8]

[9] R. Austein and M. Larson. Dns security introduction and requirements. 01 2005.

[10] Artem Maksutov, Ilya Cherepanov, and Maksim Alekseev. Detection and prevention of dns spoofing attacks. pages 84–87, 04 2017.

[11] Kamal Alieyan, Mohammad Kadhum, Mohammed Anbar, Shafiq Rehman, and Naser Alajmi. An overview of ddos attacks based on dns. 10 2016.

[12] Yunhong Gu and Robert Grossman. Abstract udt: Udp-based data transfer for high-speed wide area networks. *Computer Networks*, 51:1777–1799, 05 2007.

[13] Ye Tian, Kai Xu, and Nirwan Ansari. Tcp in wireless environments: Problems and solutions. *Communications Magazine, IEEE*, 43:S27 – S32, 04 2005.

[14] Nevil Brownlee, K.C. Claffy, and Evi Nemeth. Dns measurements at a root server. pages 1672 – 1676 vol.3, 02 2001.

[15] Xuebiao Yuchi, Wang Xin, Li Xiaodong, and Yan Baoping. Dns measurements at the .cn tld servers. pages 540 – 545, 09 2009.

[16] Moritz Müller, Giovane Moura, Ricardo Schmidt, and John Heidemann. Recursives in the wild: engineering authoritative dns servers. pages 489–495, 11 2017.

[17] Rami Al-Dalky and Kyle Schomp. *Characterization of Collaborative Resolution in Recursive DNS Resolvers*, pages 146–157. 03 2018.

[18] Francesca Meneghello, Matteo Calore, Daniel Zucchetto, Michele Polese, and Andrea Zanella. Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices. *IEEE Internet of Things Journal*, PP:1–1, 08 2019.

[19] Amir Herzberg and Haya Shulman. Dnssec: Security and availability challenges. pages 365–366, 10 2013.

[20] I. Dissanayake. Dns cache poisoning: A review on its technique and countermeasures. pages 1–6, 10 2018.

[21] Timm Böttger, Félix Cuadrado, Eder Leao Fernandes, Gareth Tyson, Ignacio Castro, and Steve Uhlig. An empirical study of the cost of dns-over-https. pages 15–21, 10 2019.

[22] Rebekah Houser, Zhou Li, Chase Cotton, and Haining Wang. An investigation on information leakage of dns over tls. pages 123–137, 12 2019.

[23] Tao Ye, Min Luo, Yi Yang, Kim-Kwang Choo, and Debiao He. A survey on redactable blockchain: Challenges and opportunities. *IEEE Transactions on Network Science and Engineering*, 10:1669–1683, 05 2023.

[24] Dejan Vujičić, Dijana Jagodic, and Siniša Ranđić. Blockchain technology, bitcoin, and ethereum: A brief overview. pages 1–6, 03 2018.

[25] Ruhi Taş and Omer Tanriover. Building a decentralized application on the ethereum blockchain. pages 1–4, 10 2019.

[26] Wen-Bin Hsieh, Jenq-Shiou Leu, and Jun-Ichi Takada. Use chains to block dns attacks: A trusty blockchain-based domain name system. *Journal of Communications and Networks*, 24(3):347–356, 2022.

[27] Jingqiang Liu, Bin Li, Lizhang Chen, Meng Hou, Feiran Xiang, and Peijun Wang. A data storage method based on blockchain for decentralization dns. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 189–196. IEEE, 2018.

[28] HU Wei-hong, AO Meng, SHI Lin, XIE Jia-gui, and LIU Yang. Review of blockchain-based dns alternatives. , 3(3):71–77, 2017.

[29] George Giamouridis, BooJoong Kang, and Leonardo Aniello. Blockchain-based dns: Current solutions and challenges to adoption. 2024.

[30] Zecheng Li, Shang Gao, Zhe Peng, Songtao Guo, Yuanyuan Yang, and Bin Xiao. B-dns: A secure and efficient dns based on the blockchain technology. *IEEE Transactions on Network Science and Engineering*, 8(2):1674–1686, 2021.

[31] Yantao Shen, Yang Lu, Zhili Wang, Xin Xv, Feng Qi, Ningzhe Xing, and Ziyu Zhao. Dns service model based on permissioned blockchain. *Intelligent Automation and Soft Computing*, 27(1):259–268, 2021.

[32] From Mallory Runyan. `https://digitalcommons.odu.edu/cgi/viewcontent.cgi?article=1014&context=covacci-undergraduateresearch#:~:text=DNS%20Root%20Servers%20or%20simply,%5B4%5D%20%5B5%5D.`

[33] Faizan Safdar Ali and Alptekin Kupcu. Improving pki, bgp, and dns using blockchain: A systematic review. *arXiv e-prints*, pages arXiv–2001, 2020.

[34] Tianfu Gao and Qingkuan Dong. Dns-bc: Fast, reliable and secure domain name system caching system based on a consortium blockchain. *Sensors*, 23(14):6366, 2023.