A

PROJECT REPORT

ON

De-centralized Blog System

SUBMITTED TO THE SHIVAJI UNIVERSITY, KOLHAPUR
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE AWARD OF DEGREE
OF

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

SUBMITTED BY

Ms. Janhavi Jaydeep Mohite	**Roll No. 116**	**20UGCS19671**
Ms. Shruti Udaykumar Katkar	**Roll No. 112**	**20UGCS19627**
Ms. Karuna Ramesh Kamble	**Roll No. 136**	**20UGCS19595**

PROJECT GROUP NO : B-16

UNDER THE GUIDANCE OF

Prof. K. N. Kamble

Department of Computer Science and Engineering
SWVSM's

Tatyasaheb Kore Institute of Engineering and Technology (Autonomous), Warananagar

Academic Year 2023-24

SWVSM's
TKIET

SWVSM's

Tatyasaheb Kore Institute of Engineering and Technology(Autonomous), Warananagar

CERTIFICATE

This is to certify that the Project Report entitled,

De-centralized Blog System

Submitted by

Name	Exam Seat No.
Ms. Janhavi Jaydeep Mohite	**20UGCS19671**
Ms. Shruti Udaykumar Katkar	**20UGCS19627**
Ms. Karuna Ramesh Kamble	**20UGCS19595**

is a bonafide work carried out and is approved for the partial fulfillment of the requirement of Shivaji University, Kolhapur for the award of Degree of Bachelor of Technology in Computer Science and Engineering. This project work is a record of students' own work, carried out by them under our supervision and guidance during academic year 2023-24.

Prof. K. N. Kamble
Project Guide

Prof. R. B. Patil
HoD, CSE

Dr. D. N. Mane
Principal

Date :-
Place :- Warananagar

External Examiner

ACKNOWLEDGEMENT

This space is dedicated to acknowledge all those who have helped in bringing this project to fruition.

We are greatly indebted to our guide ***Prof. K. N. Kamble***, for his unstinted support and valuable suggestions. We are grateful to him not only for the guidance, but also for his unending patience and keeping our spirits high throughout. We express our sincere thanks to our beloved Head of the Department, **Prof. R. B. Patil** and Principal, **Dr. S. V. Anekar** for being source of inspiration and providing us the opportunity to work on this project.

We extend heartfelt thanks to all the teaching and non-teaching staff of the department of Computer Science and Engineering for their assistance and cooperation.

Finally, we would like to thank our parents and friends for their moral support and encouragement throughout our academics.

Student Name	Exam Seat No.	Sign
Ms. Janhavi Jaydeep Mohite	**20UGCS19671**	
Ms. Shruti Udaykumar Katkar	**20UGCS19627**	
Ms. Karuna Ramesh Kamble	**20UGCS19595**	

Date :-

Place :- Warananagar

ABSTRACT

The rapid evolution of Web3 technologies offers promising prospects for decentralized and censorship-resistant applications. This project report delineates the conception, creation, and execution of a Web3 blogging system, harnessing blockchain and decentralized storage for a platform that empowers users to publish and access content securely and transparently. Commencing with an exploration of fundamental Web3 principles like blockchain and smart contracts, the report delves into an exhaustive analysis of existing decentralized blogging systems, pinpointing limitations to pave the way for a more robust solution. Subsequently, the architectural blueprint of the Web3 blogging system is elucidated, integrating Ethereum blockchain for transactions and IPFS for decentralized storage of blog posts and media assets. Smart contracts manage content ownership, user authentication, and monetization mechanisms, fostering transparency and trust. The implementation phase is meticulously outlined, encompassing smart contract development, Ethereum/IPFS integration, and the creation of a user-friendly web interface employing modern front-end frameworks. Challenges encountered during development, such as scalability and gas fees, are adeptly tackled through optimization techniques and smart contract design patterns. Moreover, the report expounds on testing methodologies employed to ensure the system's reliability and security, encompassing various testing scenarios to validate functionality and robustness. Finally, the report culminates with an evaluative overview of the Web3 blogging system, spotlighting its strengths, limitations, and potential avenues for enhancement. Exhibiting the transformative potential of Web3 technologies, this project report serves as a comprehensive guide for those intrigued by the design, development, and implementation of a Web3 blogging system, underscoring the capacity of these technologies to revolutionize content publishing in a secure and transparent manner.

Keywords: Web3 technologies, Decentralized applications, Blockchain, Smart contracts, Decentralized storage, Transparency, Security, Content publishing, Ethereum, IPFS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Name
IPFS	InterPlanetary File System
TestNet	Testing Network
E-R	Entity-Relationship
DApp	Decentralized App
HTML	HyperText Transfer Protocol
CSS	Cascading Style Sheets

Chapter 1

INTRODUCTION

1.1 Introduction

This project report focuses on the design, development, and implementation of a Web3 blogging system, leveraging blockchain and decentralized storage to create a censorship-resistant and transparent platform for content publication. The report provides an overview of the key aspects covered in the project, highlighting the objectives, methodologies, and outcomes.

Introduces the concept of Web3 and its significance in creating decentralized applications. Provides a background on the motivations behind developing a Web3 blogging system. Outlines the objectives and scope of the project. Explores the foundational technologies of Web3, such as blockchain, smart contracts, and decentralized storage. Reviews existing centralized blogging platforms and their limitations regarding censorship and control. Identifies the requirements and challenges of building a Web3 blogging system. Presents the architectural design of the Web3 blogging system, highlighting the integration of blockchain and decentralized storage. Describes the use of Ethereum blockchain for transactional operations and smart contract management. Discusses the implementation of decentralized storage solutions, such as IPFS, for storing blog posts and media assets securely.

Details the development process of the Web3 blogging system, including the creation of smart contracts and their deployment on the Ethereum network. Explores the integration of Ethereum and IPFS libraries for seamless interaction with the system. Describes the development of the user interface, emphasizing user experience and accessibility. Outlines the testing methodologies employed to ensure the reliability and security of the Web3 blogging system. Covers different types of testing conducted, including functional testing, performance testing, and security auditing. Discusses the resolution of identified issues and challenges during the testing phase.

Evaluates the Web3 blogging system based on its functionality, usability, and security. Discusses the system's strengths and limitations, considering factors such as scalability and transaction costs. Identifies potential future enhancements and areas for improvement to enhance the system's performance and user experience. Summarizes the key findings and achievements of the project. Highlights the significance of a Web3 blogging system in promoting censorship resistance and content ownership. Concludes with a call to action for continued exploration and advancement of Web3 technologies in the field of content publication.

The project report provides a comprehensive overview of the design, development, and implementation of a Web3 blogging system, emphasizing the advantages of decentralization and transparency in content publishing. It aims to contribute to the growing body of knowledge and innovation in the Web3 space while encouraging further research and development in this area.

1.2 Motivation:

Following are some points that motivates us:

Censorship Resistance:

Traditional blogging platforms are often susceptible to censorship, where content can be taken down or modified without the author's consent. A Web3 blogging system aims to address this issue by leveraging blockchain technology and decentralized storage, ensuring that content remains immutable and resistant to censorship attempts.

Content Ownership:

Many centralized blogging platforms retain ownership and control over user-generated content. In contrast, a Web3 blogging system empowers users by providing them with full ownership and control of their content. Smart contracts and blockchain technology enable verifiable proof of ownership and ensure that content creators have complete authority over their published works.

Transparency and Trust:

With Web3 technologies, transparency and trust are enhanced within the blogging ecosystem. Blockchain's decentralized nature allows for transparent and auditable transactions, ensuring that users can trust the authenticity and integrity of published content, engagement metrics, and monetization processes.

Decentralized Storage:

Traditional blogging platforms often rely on centralized servers, making them vulnerable to data breaches and downtime. By utilizing decentralized storage systems like IPFS, a Web3 blogging system ensures that content is distributed across a network of nodes, increasing data resilience, availability, and eliminating single points of failure.

Resilience against DDoS Attacks:

Distributed Denial-of-Service (DDoS) attacks can render centralized platforms inaccessible, impacting content availability and user experience. By leveraging decentralized technologies, a Web3 blogging system can distribute content across multiple nodes, making it resistant to DDoS attacks and ensuring continuous access to published content.

Data Privacy:

Centralized platforms often collect and monetize user data, raising concerns about privacy and user consent. With a Web3 blogging system, users can have greater control over their personal data. By implementing privacy-focused features, such as user-controlled identity systems or zero-knowledge proofs, a Web3 blogging system can prioritize data privacy and empower users to decide how their information is shared.

Innovation and Experimentation:

Web3 technologies provide a fertile ground for experimentation and innovation. By documenting the design and development process of a Web3 blogging system, the project report can inspire other developers and researchers to explore new possibilities, pushing the boundaries of decentralized content publishing further.

Future Impact:

Web3 technologies have the potential to reshape various industries, and content publishing is no exception. By showcasing the benefits and challenges of a Web3 blogging system, the project report contributes to the ongoing conversation around the transformative power of Web3 and its potential impact on the future of content creation, dissemination, and ownership.

1.3 Problem Statement:

Development of de-centralized blog system that resolves limitations of centralized blog system such as censorship, lack of content ownership , and vulnerabilities to dada breaches.

1.4 Objective:

Design a Decentralized Blogging System:

Present a comprehensive architectural design for a Web3 blogging system that incorporates blockchain, smart contracts, and decentralized storage to address the identified challenges and meet the requirements of a censorship-resistant and transparent platform.

Implement the Web3 Blogging System:

Describe the development process, including the creation of smart contracts, integration with Ethereum blockchain and IPFS for storage, and the implementation of a user-friendly web interface. Discuss any optimizations or design patterns used to enhance scalability and reduce gas fees.

Test and Ensure System Reliability:

Define testing methodologies and conduct various tests, such as functional testing, performance testing, and security auditing, to validate the functionality, reliability, and security of the Web3 blogging system.

Identify Future Enhancements:

Identify potential areas for improvement and enhancements, such as scalability solutions, user experience refinements, integration with emerging Web3 standards, or exploring advanced privacy features, to further enhance the functionality and effectiveness of the Web3 blogging system.

Contribute to the Web3 Ecosystem:

Foster awareness and understanding of Web3 technologies' potential and their impact on content publishing. Contribute to the ongoing discourse on Web3 applications and encourage further research and development in the field of decentralized blogging systems.

By achieving these objectives, the project report aims to provide a comprehensive resource for individuals interested in building or understanding Web3-based blogging systems, promoting the adoption and advancement of Web3 technologies in the content publishing domain.

Chapter 2

LITERATURE SURVEY

2.1 Existing System

2.1.1 Referred Journal/Conference Papers

1. Research Paper: A Blockchain based Autonomous Decentralized Online Social Network | IEEE Conference Publication | IEEE Xplore
2. Infura Doc: https://docs.infura.io/infura/
3. IPFS Doc: https://docs.ipfs.tech/concepts/what-is-ipfs/
4. React Doc: https://legacy.reactjs.org/docs/getting-started.html
5. Solidity Doc: https://docs.soliditylang.org/en/v0.8.20/
6. Web3 Doc: https://ethereum.org
7. Ethereum Doc: https://learnweb3.io

2.1.2 Elaborate on Existing System Applications / Examples

The existing system study section of the project report focuses on analyzing the current landscape of centralized blogging platforms and their limitations. It provides a critical evaluation of the shortcomings and challenges faced by these platforms, highlighting the need for a decentralized Web3-based solution. The study encompasses the following key aspects:

Centralized Blogging Platforms:

Identify and examine popular centralized blogging platforms, such as WordPress, Medium, or Blogger, Twitter, discussing their features, functionalities, and market dominance.

2.1.3 Limitations or Challenges in Existing System:

Censorship Vulnerability:

Centralized blogging platforms are susceptible to censorship, where content can be removed, modified, or blocked without the author's consent or due process. This restricts freedom of speech and limits access to diverse opinions and ideas.

Lack of Content Ownership:

Content creators on centralized platforms often relinquish ownership and control over their work. The platform retains rights to the content, making it difficult for creators to assert their ownership, enforce copyrights, or have control over monetization opportunities.

Trust and Transparency Issues:

Centralized platforms lack transparency in content moderation, engagement metrics, and monetization processes. Users cannot verify the authenticity or integrity of content, leading to trust issues within the platform.

Data Security and Privacy Risks:

Centralized platforms are vulnerable to data breaches, exposing users' personal information and compromising their privacy. Users have limited control over how their data is collected, stored, and monetized by the platform and third parties.

Scalability and Performance Limitations:

Centralized platforms may struggle to handle a large user base, leading to slower load times, downtime, and reduced user experience. As the platform grows, scalability becomes a significant challenge.

Dependency on Centralized Infrastructure:

Centralized platforms rely on centralized servers and infrastructure, making them vulnerable to single points of failure and downtime. This affects the availability and accessibility of published content.

By highlighting these limitations, the project report emphasizes the need for a decentralized Web3 blogging system that addresses these challenges, providing a more censorship-resistant, transparent, and user-centric platform for content creation and publication.

2.2 Proposed System with Block Diagram:

Specifically, the problem statement for the Web3 blog system project can be summarized as follows:

Censorship: Traditional blogging platforms are susceptible to censorship, where content can be removed, modified, or blocked without the author's consent or due process. This hampers freedom of speech and restricts access to diverse opinions and ideas.

Content Ownership: Current blogging platforms often retain ownership and control over user-generated content. This limits the rights and control of content creators, who may face challenges in asserting their ownership, enforcing copyrights, or managing monetization opportunities.

Trust and Transparency: Centralized blogging platforms lack transparency in content moderation, engagement metrics, and monetization processes. Users often cannot verify the authenticity or integrity of content, leading to trust issues within the platform.

Data Security and Privacy: Centralized platforms are vulnerable to data breaches, exposing users' personal information and compromising their privacy. Users have limited control over how their data is collected, stored, and monetized.

Scalability and Performance: With the increasing popularity of blogging platforms, scalability and performance become critical factors. Centralized systems face challenges in handling a large user base, resulting in slower load times, downtime, and reduced user experience.

Addressing these problems requires the development of a Web3 blogging system that leverages blockchain technology, decentralized storage, and smart contracts to provide a censorship-resistant, transparent, and user-centric platform. Such a system would empower users with content ownership, enhance trust and transparency, prioritize data security and privacy, and ensure scalability and performance to accommodate a growing user base.

Block Diagram:

We are going to develop this De-Centralized Blog System. The System will consist of Front End, Back End, Provider and Blockchain. The block diagram of modules is given below: The first component of block diagram is front end which is made up of JavaScript, HTML, CSS, JavaScript and React. Then the second component Provider which will do transaction on behalf of user. The Polygon is scaling solution to increase the efficiency and to reduce the transaction fee. Blog will be post on Ethereum Blockchain.

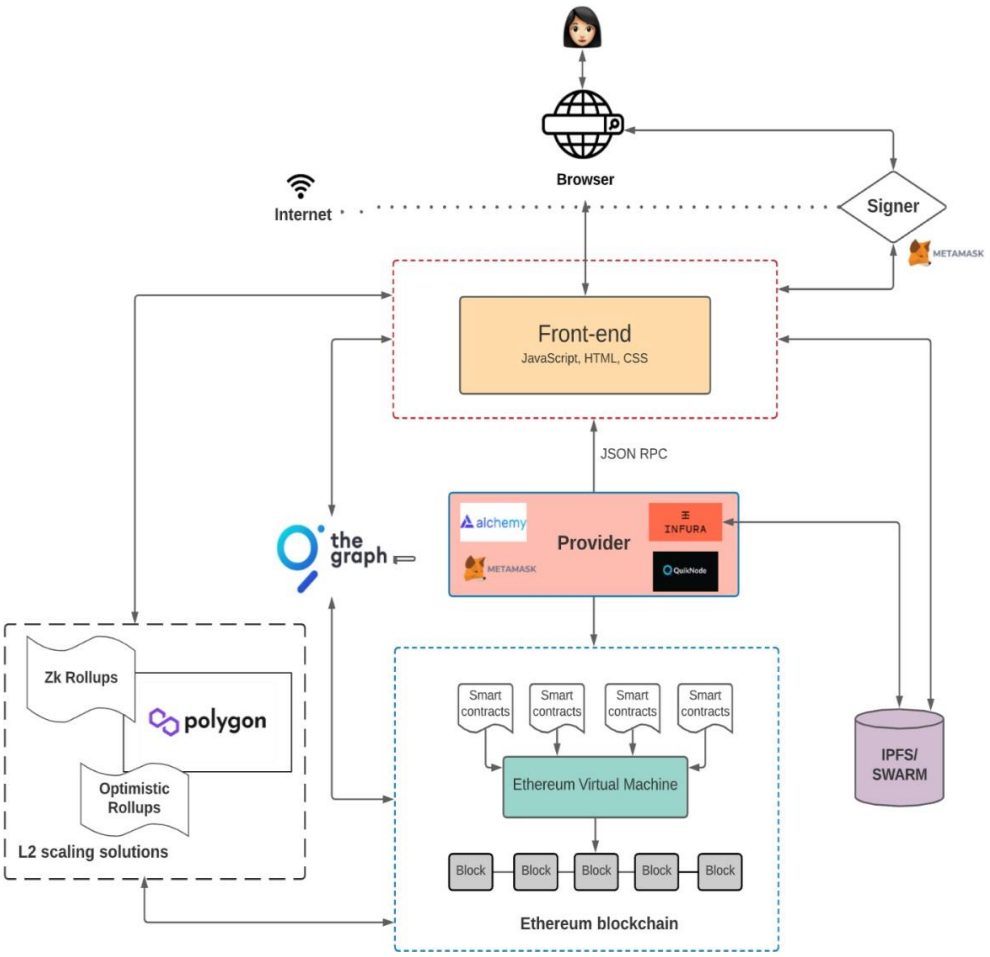

Figure 2.3: Block Diagram

Browser: The browser serves as the user's interface to access and interact with web applications. Users access a web application that incorporates Web3 functionality using a compatible browser.

Signer (Metamask): Metamask is a popular browser extension that acts as a Web3 wallet and provides a user-friendly interface for managing Ethereum accounts. It securely stores private keys and signs transactions on behalf of the user. Metamask acts as the bridge between the web application and the Ethereum blockchain.

Front-end: The front-end represents the user interface and user experience (UI/UX) of a web application. It typically comprises HTML, CSS, and JavaScript code that interacts with the user and communicates with the back-end.

Provider: The provider is a software component that connects the front-end of the web application with the Ethereum blockchain. It enables communication between the web application and the Ethereum network, allowing interaction with smart contracts and data retrieval. The provider can be a library like Web3.js or Ethers.js, which facilitates the integration between the front-end and the Ethereum blockchain.

Smart Contract: A smart contract is a self-executing contract stored on the Ethereum blockchain. It contains the business logic and rules that govern the behavior of a decentralized application (DApp). Smart contracts are written in Solidity or other compatible programming languages.

Ethereum Blockchain: The Ethereum blockchain is a decentralized, distributed ledger that records transactions and executes smart contracts. It serves as the underlying infrastructure for decentralized applications built on the Ethereum platform. The blockchain stores the state of the Ethereum network and processes transactions based on the consensus mechanism.

IPFS (InterPlanetary File System): IPFS is a peer-to-peer distributed file system that enables decentralized storage and retrieval of files. It provides content-addressable storage, allowing files to be uniquely identified by their content, which enhances data integrity and availability. IPFS can be used to store and retrieve large files, such as media or documents, in a decentralized manner.

In this architecture, the flow typically involves the following steps:

- The user interacts with the front-end of the web application through the browser.
- The front-end communicates with the provider, such as Web3.js, to send requests and retrieve data from the Ethereum blockchain.
- When a user wants to perform a transaction, the front-end triggers the Metamask signer to prompt the user for approval.
- Metamask signs the transaction using the user's private key and relays it back to the front-end.
- The front-end broadcasts the signed transaction to the Ethereum network via the provider.
- Miners on the Ethereum network validate the transaction, include it in a block, and add the block to the blockchain

2.3 Feasibility Study -

2.3.1 Technical Feasibility –

- The Project is web application based on de-centralized blockchain.
- The Main Technologies and tools are associated-
  1. Front-End Language : React
  2. Programming Language : Solidity
  3. IPFS & Provider : Infura
  4. Blockchain : Ethereum, Goerli Testnet, Ganache
  5. IDE : Visual Studio Code
  6. Diagramming Tools : Draw.io, MS word office
- Most of the technologies are freely available and required technical skills are manageable initially website will be run locally.
- By considering above points we can conclude that our project is technically feasible.

2.3.1 Legal Feasibility –

- The project is about developing a De-centralized Blog System using Ethereum Blockchain.
- This project is legal and doable.
- It meets all the legal and ethical requirements as per IT ACT 2000, Govt. Of India.
- No threat to clients.

Hence, it can be concluded that project is legally feasible with no potential infringement**.**

Chapter 3

PROJECT SCOPE AND REQUIREMENT ANALYSIS

3.1 Project Scope:

The project scope defines the boundaries and objectives of the Web3 blog system project report. It outlines the specific areas and components that will be covered in the report. The scope for the project report include the following.

3.1.1 In-scope:

System Architecture: Provide a detailed description of the architectural design of the Web3 blog system, including the integration of blockchain technology, smart contracts, decentralized storage (such as IPFS), and user interfaces.

User Authentication: Discuss the mechanisms for user authentication, and identity management within the Web3 blog system. Explore decentralized identity solutions and their integration with the system.

Content Creation and Publishing: Explain the process of content creation and publishing on the Web3 blog system. Discuss the features and functionalities available to users, including text formatting, media embedding, tagging, and categorization.

Performance and Scalability: Address the performance and scalability considerations of the Web3 blog system. Discuss strategies for optimizing system performance, managing network congestion, and ensuring efficient resource utilization.

The project scope should be defined in a way that ensures a comprehensive and focused exploration of the key components of the Web3 blog system. It should encompass the core functionalities and considerations relevant to the development and deployment of a decentralized blogging platform using Web3 technologies.

3.1.2 Out-of-scope:

Detailed Cryptography and Blockchain Concepts:

While it is important to have a high-level understanding of the underlying technologies, the project report should not delve into intricate details of cryptography algorithms or the inner workings of blockchain protocols.

Development of Web3 Frameworks or Libraries:

The project report should not include the development of Web3 frameworks or libraries from scratch. Instead, it should focus on utilizing existing Web3 frameworks and tools to implement the blogging system.

Implementation of a Full-Featured Blogging Platform:

While the project may involve the development of a functional prototype or proof-of-concept, the project report should not aim to cover all the complexities and features of a fully matured, production-ready blogging platform.

Deep Dive into Specific Blockchain Networks:

The report should not extensively discuss the technical details or implementation specifics of a particular blockchain network, such as Ethereum. The focus should be on the conceptual integration of Web3 technologies.

User Interface Design and Styling:

The report should not extensively cover the user interface design and styling aspects of the Web3 blog system. While the basic user interface elements and interactions can be discussed, the detailed design and aesthetics are out of scope.

The out of scope items should be clearly defined to ensure that the project report remains focused on the key components and objectives of the Web3 blog system, avoiding unnecessary tangents or areas that are beyond the scope of the project.

3.1.3. Project boundaries: -

System Development and Implementation:

The project report should focus on the conceptual design, development, and implementation of the Web3 blog system. It should cover the technical aspects of integrating Web3 technologies, such as blockchain and smart contracts, into the blogging system.

Integration with Specific Web3 Technologies:

The project report should primarily focus on integrating the Web3 blog system with specific Web3 technologies, such as a particular blockchain network (e.g., Ethereum) and decentralized storage solutions (e.g., IPFS). Integrations with alternative technologies or multiple blockchain networks may be considered out of scope.

Limited Scalability Considerations:

While scalability is important, the project report may not extensively cover large-scale performance testing or implementation strategies for handling a massive user base. The focus should be on discussing scalability challenges within the context of the project's scope and constraints.

Technical Implementation Details:

The project report should provide an overview of the technical implementation of the Web3 blog system but should not delve into low-level programming details or specific code snippets. It should focus on the overall architecture, design patterns, and integration methodologies. Focus should be on the technical implementation of the Web3 blog system.

Setting clear project boundaries helps ensure that the project report remains focused on the key aspects of developing a Web3 blog system. It allows for a thorough exploration of the technical integration of Web3 technologies while excluding unrelated or complex areas that are beyond the project's scope.

3.2 Requirement Gathering:

User Registration and Authentication: Users should be able to create accounts and register on the website. The website should have a robust authentication system to ensure secure access to user accounts.

Blog Post Creation: Users should be able to create new blog posts. The post creation process should include fields for the title, content, and any relevant tags or categories. Users may also have the option to add images, videos, or other media to their blog posts.

Publishing and Visibility: Once a user creates a blog post, they should have the option to publish it. Published blog posts should be visible to all users of the website. Users should not be able to edit their blog posts after publishing to maintain the integrity of the content.

Security and Privacy: The website should implement necessary security measures to protect user data. User privacy should be respected, and personal information should be handled securely

Mobile Responsiveness: The website should be accessible and provide a good user experience on various devices, including mobile phones and tablets

User Wallet Integration: Users should have a web3 wallet associated with their account. The website should integrate with a blockchain platform (e.g., Ethereum) to enable wallet functionality. Users should be able to connect their wallet to the website securely. The wallet should support the storage and management of cryptocurrencies or tokens relevant to the web3 ecosystem.

3.3 Requirement Analysis:

3.3.1 Functional Requirements:

The main motive of this project is to share thoughts in the form of text, image through blogs and store these blogs in the de-centralized manner for more security and authority. To provide login and registration facility to users, search option to find any topic related blog and attractive interface to write the blog.

1) Login –

a) Validity Checks –

- Every user will have a unique login Id.
- Cryptographic signature validation.
- First user should be login into metamask wallet.
- User can directly login through metamask button.
- User can also login by scanning QR code for this user should be login into metamask wallet on their mobile.

b) Response to abnormal Situations –

If the flow of any of the validations does not hold true, an appropriate error message will be prompted to the user for doing the needful.

2) Create Post –

a) Validity Checks –

- User should be login into the system.
- Title and Blog should not be empty.
- URL and Image URL should be proper and unbroken.

b) Response to abnormal Situations –

If the flow of any of the validations does not hold true, an appropriate error message will be prompted to the user for doing the needful.

3) Read Post –

a) Validity Checks –

- User can read content without login into the system.
- Blog should not be empty.
- Blog content should be visible properly.

b) Response to abnormal Situations –

If the flow of any of the validations does not hold true, an appropriate error message will be prompted to the user for doing the needful.

3.3.2 Non-Functional Requirements:

1. **Performance:**

   The system provides the high performance for result generation so that system can work properly. Because many number of miners are available to mine that transaction on blockchain. Blockchain can handle many transaction at a time.

2. **Reliability:**

   The system is reliable so that user can use the system properly. On blockchain miners are mining 24x7. So, user can use system and do transaction at any time.

3. **Maintainability:**

   The system is easy to maintain if any changes or failure is occurred. Gas fee for the transaction is the only maintenance for posting blogs.

4. **Usability:**

   The user can easily use the system at any time because of the proper reliability of the system.

Chapter 4

PROJECT DESIGN AND MODELING DETAILS

4.1 Software Requirement Specification (SRS):

This project report outlines the development of a Web3 blogging system, leveraging blockchain and decentralized storage for secure, transparent content publishing. It explores core Web3 concepts like blockchain and smart contracts, contrasting with existing decentralized systems to inform a robust solution. The architectural design integrates Ethereum blockchain for transactions and IPFS for decentralized storage, with smart contracts managing content ownership and user authentication. Implementation details cover smart contract development, library integration, and front-end design, addressing scalability and gas fee challenges. Testing methodologies ensure reliability and security, validating functionality and robustness. The report concludes with an evaluation, emphasizing the system's strengths and potential improvements. It highlights Web3's capacity for decentralized, censorship-resistant content publishing, offering insights for those interested in Web3 system development.

Functional Requirements:

1) **Login –**

a) Validity Checks –

i) Every user will have a unique login Id.

ii) Cryptographic signature validation.

iii) First user should be login into metamask wallet.

iv) User can directly login through metamask button.

v) User can also login by scanning QR code for this user should be login into metamask wallet on their mobile.

b) Response to abnormal Situations –

If the flow of any of the validations does not hold true, an appropriate error message will be prompted to the user for doing the needful.

2) Create Post –

a) Validity check –

i) User should be login into the system

ii) Title and Blog should not be empty.

iii) URL and Image URL should be proper and unbroken.

b) Response to abnormal Situations –

If the flow of any of the validations does not hold true, an appropriate error message will be prompted to the user for doing the needful.

3) Read Post –

a) Validity Checks –

i. User can read content without login into the system.

ii. Blog should not be empty.

iii. Blog content should be visible properly.

b) Response to abnormal Situations –

If the flow of any of the validations does not hold true, an appropriate error message will be prompted to the user for doing the needful.

Non-Functional Requirements:

1. Performance :

The system provides the high performance for result generation so that system can work properly.

2. Reliability :

The system is reliable so that user can use the system properly.

3. Maintainability :

The system is easy to maintain if any changes or failure is occurred.

4. Usability :

The user can easily use the system on any time.

4.2 System Modules:

Below are the modules on which we are going to work –

- **Front End Development:**

  We are going to use React for developing our front end. We need to connect our front end to any kind of crypto wallet for that we are using Ethers library.

  We are using following libraries for the front end –

  1) useState from 'React 2) Ethers from 'ethers' 3) css from '@emotion/css'
  2) easymde/dist/easymde.min.css

- **Login:**

  Provide a "Connect" button for the login. Once you click on the "Connect" button, it should prompt your MetaMask wallet, assuming you have the browser extension installed. The backend provides a random message for the user to sign in with his or her private key in the MetaMask wallet. The signed message is then returned to the backend, together with the user's public Ethereum address.

  We are using following libraries for the login –

  1) Web3Modal 2) @walletconnect/web3-provider

- **Create Posts:**

  Provide a Text Editable interface for the user to write a blog using React. We need to create a blog post schema that includes fields such as title, content, cover photo. Use tools like IPFS to store your data. Once the user submits the content, we are using Infura Provider to interact with the Ethereum blockchain and store the blog post data on the blockchain.

  We are using following libraries for the create post –

  1) Ethers 2) ipfs-http-client 3) react-simplemde-editor

- **Read Posts:**

Once the user submits the content, we are using Infura Provider to interact with the Ethereum blockchain and store the blog post data on the blockchain. We can display the blog posts on website using a tool like IPFS which is a peer-to-peer network for storing and sharing hypermedia in a distributed file system. First it will get the hash value of the blog stored on the blockchain. Then using this parameter the data stored on the IPFS will be fetch.

We are using following libraries for the read post – 1) Ethers

- **Logout:**

The Logout module is an essential component of the Web3 Blog System, allowing users to securely log out of their accounts. It ensures the protection of user data and prevents unauthorized access to their accounts.

We are using following libraries for the logout - 1) useState from 'React'

- **Provider – Meta Mask:**

MetaMask allows users to store and manage account keys, broadcast transactions, send and receive Ethereum-based cryptocurrencies and tokens, and securely connect to decentralized applications through a compatible web browser or the mobile app's built-in browser. Websites or other decentralized applications are able to connect, authenticate, and/or integrate other smart contract functionality with a user's MetaMask wallet via JavaScript code that allows the website to send action prompts, signature requests, or transaction requests to the user through MetaMask as an intermediary.

Learning Resource - https://www.youtube.com/watch?v=gfD_kuoZt-Q

- **Ethereum/Goerli blockchain:**

Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts. Smart contracts allow participants to transact with each other without a trusted central authority. **Ganache** is a popular personal blockchain for Ethereum development. It allows developers to create a local Ethereum network, which can be used for testing, debugging, and building decentralized applications (dApps) without the need for real Ether or connecting to the main Ethereum network.

Learning Resource - https://www.youtube.com/watch?v=UnNPv6zEbwc

4.3 System Modeling and Design

The system is separated into various parts and designs. Following diagrams will explain the design:

4.3.1 Class Diagram

The system main contains the classes such as user, login, blog. The classes will contains methods like login() and logout(), Write_data() and Read_data().

Figure 4.3.1: Class Diagram

4.3.2 Sequence Diagram

The sequence of the above system is explained below. The system diagram shows that the sequence start with user after login and end on logout. The following classes will perform intermediate operations.

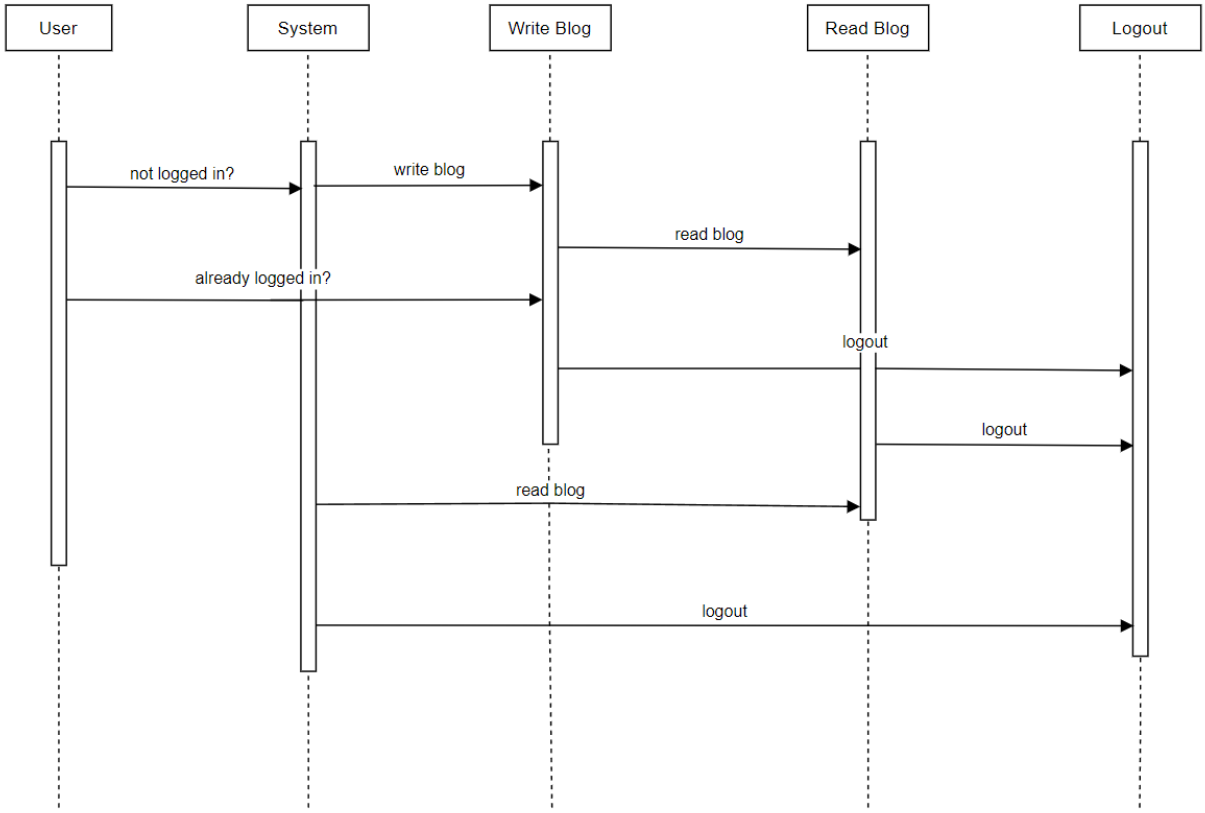

Figure 4.3.2: Sequence Diagram

4.3.3 Activity Diagram

The activity of above proposed system is shown below. The activity have two ends – write blog and read blog. User can first login into the system. To write a blog login is mandatory but for reading the blogs the login condition is not mandatory. Blog will be published on blockchain and stored on IPFS. The hash value of blog will be stored on blockchain.

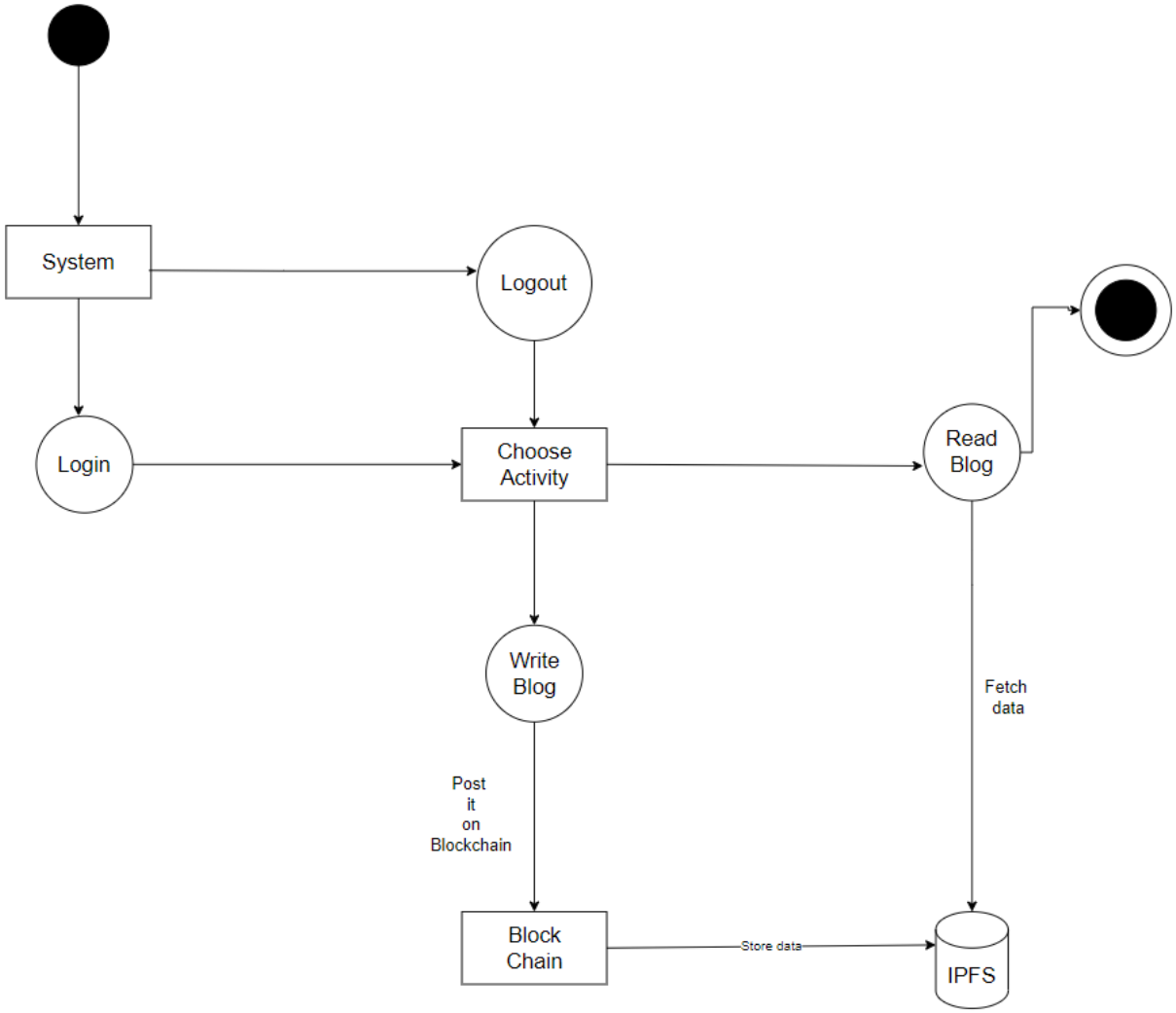

Figure 4.3.3: Activity Diagram

4.3.4 Use case Diagram

The use case diagram is shown below. First user will perform login. Afterwards he will perform different actions related to blog like reading or writing blogs .

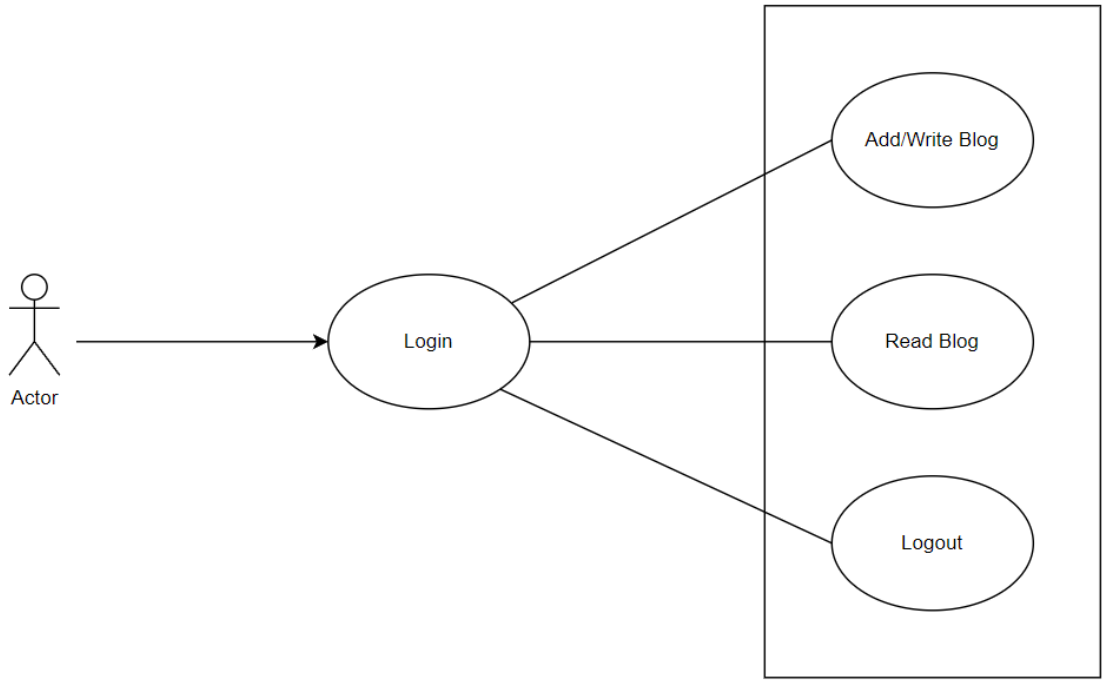

Figure 4.3.4: Use Case Diagram

4.4 Database Design

4.4.1 ER-Diagram

The Entity-Relationship diagram show the relationship between the user and the blog table. One user will publish a blog on blockchain. The hash value will be stored on blockchain. If the another user wants to read that blog, that blog will be shown by using the hash value stored on blockchain.

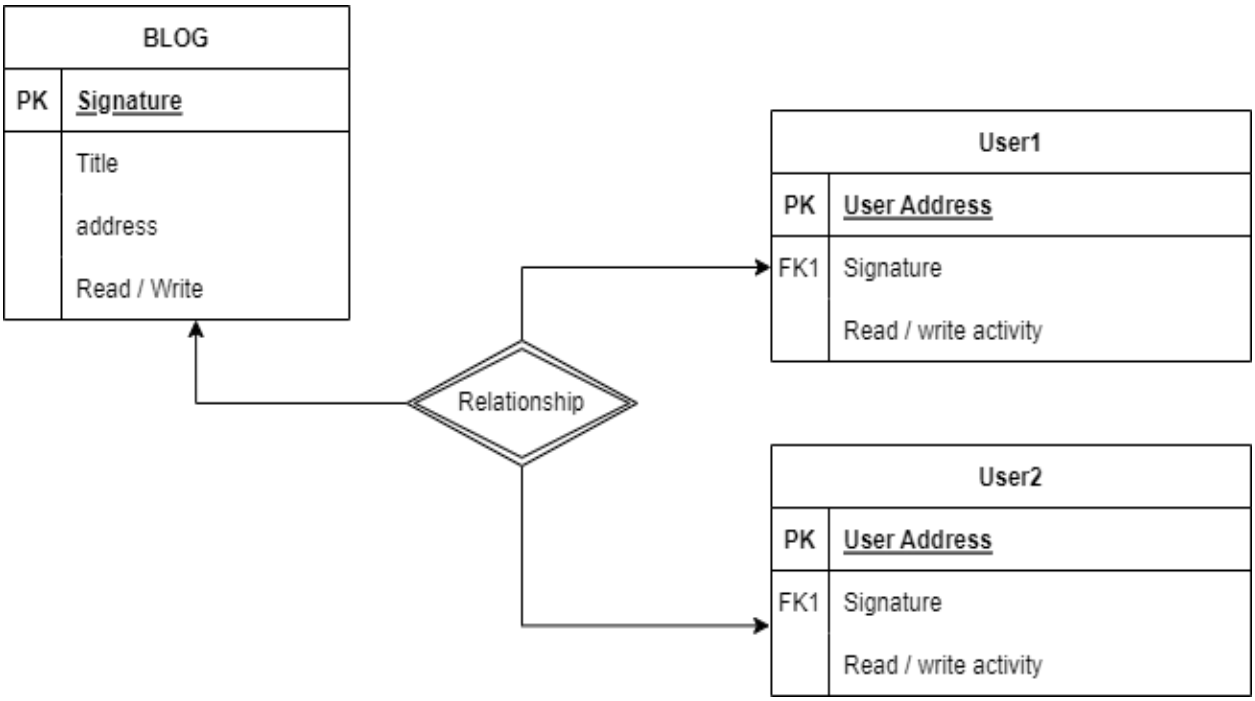

Figure 4.4.1: ER-Diagram

4.5 System Architecture:

The architecture of a Decentralized Blog System comprises various layers and components, each playing a crucial role in enabling decentralized content creation, storage, and sharing:

1. User Interface Layer: User-facing components like web or mobile applications facilitate interactions such as blog creation, post management, and content discovery.

2. Application Layer: Core logic and functionalities, including user authentication, blog management, content interaction, and integration with blockchain technology, reside here.

3. Blockchain Layer: Foundation of decentralization, including smart contracts for functionality, content storage on the blockchain or IPFS, and consensus mechanisms for transaction validity.

4. Decentralized Network Layer: Peer-to-peer network enabling decentralized data exchange, managed by node software and P2P communication protocols.

5. Data Storage Layer: Utilizes decentralized storage solutions like IPFS or decentralized databases for efficient handling of content storage.

6. External Services: Integration points for functionalities such as email notifications, analytics, and social media sharing.

7. Integration Layer: Handles integration with external systems or APIs for transactions, identity verification, and content indexing.

8. Security Layer: Integrates security measures like encryption, access control, and auditing to protect user data and ensure transaction integrity.

9. Monitoring and Analytics: Tools for monitoring system health, performance metrics, and user behavior analytics aid in system optimization.

10. Governance Mechanisms: Enables user participation in platform development, content moderation, and community management decisions.s

Chapter 5

Implementation and Coding

5.1 Algorithms:

1. User Registration and Authentication:
- Users register accounts with unique identifiers and credentials.
- Authentication mechanisms verify user identity securely.

2. Content Creation:
- Users create blog posts, including text, images, and other media.
- Content creation tools enable formatting and editing capabilities.

3. Decentralized Storage:
- Blog posts are encrypted and stored in a decentralized storage network, such as IPFS (InterPlanetary File System).
- Each post receives a unique identifier (CID) for retrieval and verification.

4. Blockchain Integration:
- Utilize a blockchain network, such as Ethereum, for transactional operations and smart contract management.
- Smart contracts govern user interactions, content ownership, and monetization mechanisms.

5. Posting Content:
- Users submit blog posts to the system, triggering a transaction on the blockchain.
- Smart contracts verify ownership and store post metadata, including author, timestamp, and content CID.

6. Content Discovery:
- Users explore and discover blog posts through a decentralized indexing mechanism.
- Indexing services retrieve post metadata from the blockchain and provide search and filtering capabilities.

7. Interaction and Engagement:
- Smart contracts manage interactions securely, updating post metadata accordingly.

8. Moderation and Governance:
   - Administrators monitor and moderate user-generated content for compliance with community guidelines.
   - Governance mechanisms enable community voting and decision-making processes.

9. Monetization:
   - Introduce monetization features, such as micro-payments or subscriptions, facilitated by smart contracts.
   - Content creators earn rewards or payments based on engagement metrics or user contributions.

10. Decentralized Identity:
   - Users control their identity and personal data using decentralized identity solutions.
   - Identity management systems ensure privacy, security, and user autonomy.

11. Security and Auditing:
   - Implement security measures to protect user data and system integrity.
   - Conduct regular audits and security assessments to identify and mitigate vulnerabilities.

12. Scalability and Optimization:
   - Continuously optimize system performance and scalability through protocol upgrades and network enhancements.
   - Implement scaling solutions, such as sharding or layer 2 protocols, to accommodate increasing user demand.

13. User Interface and Experience:
   - Design and develop a user-friendly interface for seamless interaction with the decentralized blog system.
   - Prioritize accessibility, responsiveness, and intuitive navigation.

14. Testing and Quality Assurance:
   - Conduct comprehensive testing, including unit tests, integration tests, and user acceptance tests.
   - Ensure compliance with functional requirements and performance benchmarks.

15. Documentation and Support:
   - Provide comprehensive documentation for system users, developers, and **administrators.**
   - Offer ongoing support and community resources for troubleshooting and assistance.

5.2 Development Requirements:

a) Software Requirement:

Operating system	:	Windows7 or Higher
Front design	:	React
Programming language	:	Solidity
Database	:	IPFS

b) Hardware Requirement:

Processor	:	Pentium IV or Higher
Hard Disk	:	1GB
RAM	:	512MB or more

5.3 Deployment Requirements:

c) Software Requirement:

Operating system	:	Windows7 or Higher
Front design	:	React
Programming language	:	Solidity
Database	:	IPFS

d) Hardware Requirement:

Processor	:	Pentium IV or Higher
Hard Disk	:	1GB
RAM	:	512MB or more

Chapter 6
Testing

6.1 Fundamentals of Testing:

1. Decentralization Testing: Ensure that the system functions properly in a decentralized environment, with nodes communicating effectively and data replication working as expected.

2. Data Integrity Testing: Verify that data stored on the decentralized network remains intact and consistent across all nodes, even in the presence of network disruptions or node failures.

3. Security Testing: Assess the system's resistance to various security threats, such as unauthorized access, data manipulation, and denial-of-service attacks, considering the distributed nature of the network.

4. Performance Testing: Evaluate the system's performance under different loads and network conditions, ensuring that it can handle a high volume of blog posts and user interactions efficiently.

5. Consensus Mechanism Testing: Test the consensus algorithm used by the decentralized network to reach agreement on the validity of blog posts and transactions, ensuring that it functions correctly and reliably.

6. User Experience Testing: Validate the user experience of interacting with the decentralized blog system, including usability, responsiveness, and accessibility across different devices and platforms.

7. Fault Tolerance Testing: Assess the system's ability to gracefully handle and recover from node failures, network partitions, and other unexpected events without compromising data integrity or availability.

8. Interoperability Testing: Verify compatibility and interoperability with other decentralized systems, protocols, and standards, ensuring seamless integration and communication between different components and networks.

6.2 Test Plan of the Project –

A test plan for a Decentralized blog system would typically include various testing activities to ensure the system functions correctly, securely, and efficiently. Here's an outline of a test plan for a Decentralized blog system:

1. Introduction:

- Provide an overview of the web3 blog system and its key features.
- Define the objectives and scope of the testing.
- Identify the testing team members and their roles.

2. Test Environment:

- Specify the hardware, software, and network requirements for the test environment.
- Set up the necessary blockchain network and infrastructure components.
- Ensure the availability of test data and sample blog posts.

3. Test Strategy:

- Define the overall approach to testing, including the types of testing to be conducted (e.g., functional, security, performance).
- Determine the testing techniques and methodologies to be used (e.g., black-box, white-box, integration testing).
- Identify any specific tools or frameworks required for testing.

4. Test Scenarios:

- Identify and document various test scenarios based on the functional requirements of the web3 blog system.
- Include scenarios for creating blog posts, editing/deleting posts, user authentication, user roles and permissions, commenting, etc.
- Consider different user roles (e.g., administrators, authors, readers) and their specific actions and permissions

5. Test Cases:

- Develop detailed test cases for each identified test scenario.

- Specify the steps to be executed, input data to be used, and expected results for each test case.

- Include both positive and negative test cases to validate system behavior under different conditions.

6. Test Execution:

- Execute the test cases based on the defined test scenarios.

- Log any defects or issues encountered during testing, including steps to reproduce and expected results.

- Perform functional testing, security testing, and performance testing as per the defined strategy.

7. Test Data Management:

- Manage test data, including creating sample blog posts, user accounts.

- Ensure the test data covers a wide range of scenarios and edge cases.

- Regularly refresh and update the test data to ensure accurate and reliable testing.

8. Test Reporting and Documentation:

- Document the test execution results, including test case status, defects found, and any deviations from expected behavior.

- Generate comprehensive test reports highlighting the overall system quality and any recommendations for improvement.

- Update the test documentation, including test cases, based on the results and feedback obtained during testing.

6.3 Test Cases and Test Results –

Module 1: Login:

Test Case	Description	Expected Output	Original Output	Result
Test Case 1	Verify MetaMask Login Button	MetaMask login button is displayed	MetaMask login button is displayed	Pass
Test Case 2 :	Verify MetaMask Wallet Connection	Successful connection to MetaMask wallet	Successful connection to MetaMask wallet	Pass
Test Case 3:	Verify Unauthorized Access Handling	Access denied	Access denied	Pass
Test Case 4:	Verify Wallet Switching	Wallet switched successfully	Wallet switched successfully	Pass
Test Case 5:	Verify Account Balance Display	Account balance is correctly displayed	Account balance is correctly displayed	Pass

Table 6.3.1 Login Module Test Cases

Module 2: Create Blog:

Test Case	Description	Expected Output	Original Output	Result
Test Case 1	Verify User Login Requirement	User must be logged in to create a blog	User must be logged in to create a blog	Pass
Test Case 2 :	User must be logged in to create a blog	Blog will not publish	Blog is not publishing	Pass
Test Case 3:	Verify Content Cannot Be Empty	Blog will not publish	Blog will not publish	Pass
Test Case 4:	Verify Wallet Has Gas Fee	Blog will not publish	Blog will not publish	Pass

Table 6.3.2 Create Blog Module Test Cases

Module 3: Read Blog:

Test Case	Description	Expected Output	Original Output	Result
Test Case 1	It's Okay If User Is Not Logged In	User can read blogs without logging in	User can read blogs without logging in	Pass
Test Case 2 :	Contract Should Return Valid Hash ID	Valid hash ID returned from the contract	Valid hash ID returned from the contract	Pass
Test Case 3:	IPFS Project Should Be Running	IPFS project is running and blogs are accessible	IPFS project is running and blogs are accessible	Pass

Table 6.3.3 Read Blog Module Test Cases

Module 4: Logout:

Test Case	Description	Expected Output	Original Output	Result
Test Case 1	Successful Logout	User successfully logged out	User successfully logged out	Pass
Test Case 2 :	Logout Redirect	Redirect to the login page	Redirect to the login page	Pass
Test Case 3:	Session Termination	User session terminated	User session terminated	Pass
Test Case 4:	Invalidation of Authentication Tokens	Authentication tokens invalidated	Authentication tokens invalidated	Pass
Test Case 5:	Logout Button Visibility	Logout button is visible	Logout button is visible	Pass

Table 6.3.4 Logout Module Test Cases

Chapter 7
Project Plan and Schedule

7.1 Project planning and project resources

Task	Description	Duration
Domain Knowledge	Discussion with guide about different problems	40 days
Problem Definition	Selection of project and problem statement with guide and client	20 days
Requirement Analysis	Client meeting	10 days
	Information Gathering	10 days
Design	System Architecture	30 days
	Synopsis	15 days
Planning	DFD, UML, Activity diagram, State diagram, Use-case diagram	20 days
Initial Report	Compilation of initial report	15 days
Implementation	Initializing and finalizing implementation	20 days
System Testing	Testing different scenarios	15 days
Final Report	Compiling final report	18 days

Table 7.1.1: Project planning

7.2 Project Scheduling:

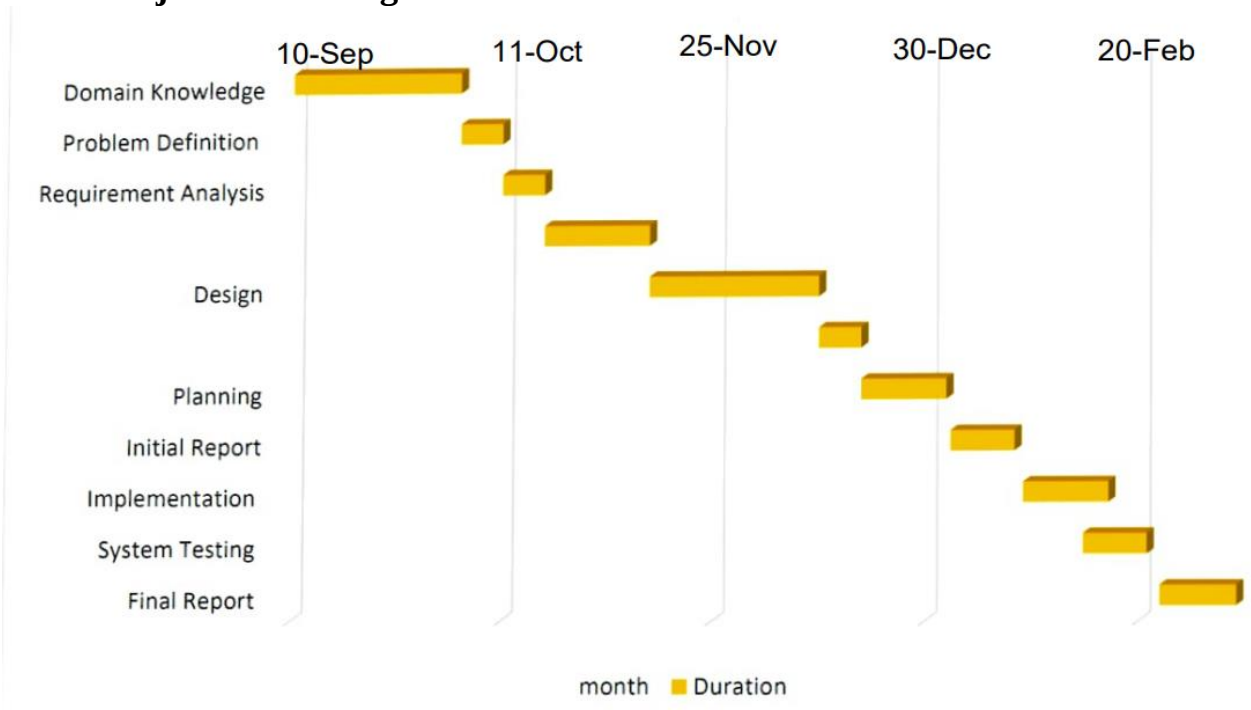

Fig 7.2.1: Gantt Chart

We created project schedule by using following steps:

Task Identification: Identify the tasks required to complete each module and the main code file. For example, tasks could include data preprocessing, algorithm implementation, testing, and integration.

Task Dependencies: Determine the dependencies between tasks. Identify which tasks need to be completed before others can start. For instance, module2.py may depend on the successful execution of module1.py.

Task Duration Estimation: Estimate the time required to complete each task. Consider factors such as complexity, the size of the code, computational requirements, and any potential external dependencies.

Task Sequencing: Based on the task dependencies and duration estimates, create a sequence of tasks. Determine the order in which the tasks should be executed to optimize the project timeline. For example, you may need to complete module1.py before starting module2.py.

Resource Allocation: Identify the resources required for each task. This includes personnel, computing resources, and any external dependencies. Ensure that the necessary resources are available when needed.

Once you have defined the project schedule, you can visualize it using a Gantt chart. A Gantt chart is a bar chart that illustrates the project schedule over time. Each task is represented by a horizontal bar, and the length of the bar corresponds to the task duration. The Gantt chart provides a visual representation of the project timeline, task dependencies, and resource allocation.

To create a Gantt chart for your project, you would list the tasks on the left side of the chart and plot the bars representing the start and end dates of each task along a horizontal timeline. You can also include dependencies and milestones to enhance the clarity of the chart. The Gantt chart helps in monitoring the progress of the project, identifying potential bottlenecks, and managing resources effectively.

Chapter 8

Risk Management and Analysis

8.1 Project Risk Identification:

Identifying risk is one of most important or essential and initial steps in risk management process. By chance, if failure occurs in identifying any specific risk, then all other steps that are involved in risk management will not be implemented for that particular risk.

1. Smart Contract Vulnerabilities: Web3 blog systems often rely on smart contracts to handle content creation, publishing, and interactions. Identifying and addressing potential vulnerabilities in smart contracts is crucial to mitigate the risk of exploitation or malicious activities.

2. Privacy and Data Security: Web3 blog systems typically store user data and content on the blockchain or decentralized storage networks. Risks related to privacy breaches, unauthorized access, and data leaks need to be identified and managed effectively.

3. Network and Infrastructure Risks: Web3 relies on a decentralized network infrastructure, including blockchain networks and peer-to-peer protocols. Risks associated with network congestion, scalability limitations, and reliance on specific networks need to be assessed to ensure smooth operations.

4. Smart Contract Upgradability: Web3 blog systems may utilize upgradable smart contracts to incorporate new features or fix bugs. However, the risks of introducing unintended behaviors or vulnerabilities during the contract upgrade process must be identified and managed appropriately.

5. Identity Management: Web3 blog systems may use decentralized identity solutions, such as self-sovereign identity or decentralized identifiers. Risks related to identity theft, impersonation, or inadequate authentication mechanisms need to be identified and addressed.

6. Community Governance Risks: Web3 blog systems often involve community governance models, where token holders or users participate in decision-making. Risks related to governance manipulation, conflicts of interest, or collusion within the community should be identified and managed.

7. Interoperability Challenges: Web3 is built on various blockchain protocols and standards, and interoperability between different platforms can pose risks. Identifying potential issues related to cross-chain transactions, data compatibility, or protocol inconsistencies is important for a seamless user experience.

8. External Data Dependency: Web3 blog systems may rely on external data sources or oracles to provide real-time information. Risks associated with the accuracy, reliability, or tampering of external data need to be assessed to ensure the integrity of the system.

10. User Experience and Adoption Risks: Web3 blog systems aim to provide a user-friendly and intuitive experience. Risks related to complexity, steep learning curves, or limited user adoption can hinder the success of the system. Identifying and addressing these risks is crucial for attracting and retaining users.

8.2 Project Risk Analysis:

1. Smart Contract Vulnerabilities:

Risk: Smart contracts may contain coding errors or vulnerabilities that can be exploited by attackers, leading to financial losses, unauthorized access, or manipulation of content.

Mitigation: Conduct thorough code audits, utilize standardized secure coding practices, perform penetration testing, and implement bug bounty programs to identify and address potential vulnerabilities in smart contracts.

2. Privacy and Data Security:

Risk: Storing user data on the blockchain or decentralized storage networks may expose sensitive information to unauthorized access or data leaks.

Mitigation: Implement encryption techniques, user-controlled access permissions, pseudonymity, and anonymization mechanisms to protect user privacy. Regularly monitor and update security measures to ensure data security.

3. Network and Infrastructure Risks:

Risk: Network congestion, scalability limitations, or reliance on specific blockchain networks can impact the performance and availability of the Web3 blog system.

Mitigation: Assess the scalability and throughput capabilities of the chosen blockchain network. Explore layer 2 solutions or alternative blockchains to mitigate scalability limitations. Develop contingency plans for network disruptions or congestion.

4. Smart Contract Upgradability:

Risk: Upgrading smart contracts can introduce unintended behaviors or vulnerabilities, potentially compromising the integrity and security of the Web3 blog system.

Mitigation: Implement a robust upgrade mechanism with thorough testing and auditing processes. Utilize multisignature approval processes or timelocks to mitigate the risk of unauthorized or hasty upgrades.

5. Community Governance Risks:

Risk: Manipulation, conflicts of interest, or collusion within the community can undermine the fairness and transparency of the governance process.

Mitigation: Establish clear governance rules, voting mechanisms, and participation requirements. Encourage transparency, accountability, and diversity within the community. Implement measures to detect and prevent governance manipulation or collusion.

thorough testing and verification when integrating with external blockchain networks.

6. External Data Dependency:

Risk: Reliance on external data sources or oracles can introduce the risk of inaccurate, manipulated, or delayed data, which may impact the accuracy and integrity of the Web3 blog system.

Mitigation: Establish trusted oracle systems or utilize reputable data providers. Implement data verification mechanisms and consensus algorithms to ensure the accuracy and reliability of external data sources.

7. User Experience and Adoption Risks:

Risk: Complex user interfaces, steep learning curves, or limited user adoption can hinder the success and growth of the Web3 blog system.

Mitigation: Prioritize user experience and design intuitive interfaces. Provide user-friendly documentation, tutorials, and educational resources to facilitate user onboarding. Continuously gather user feedback and iterate on the system to improve adoption.

Chapter 9
Configuration Management

9. Configuration Management:

Configuration management refers to the process of managing and controlling the configurations and settings of software systems. In the context of the Web3 blog system project report, configuration management plays a crucial role in ensuring the consistency, stability, and reproducibility of the system. Here are some key points to include in the project report regarding configuration management:

Configuration Items: Identify the configuration items within the Web3 blog system. This includes components such as the blockchain network, smart contracts, decentralized storage, user authentication mechanisms, and any other relevant software or infrastructure components.

Version Control: Discuss the use of version control systems, such as Git, to manage the source code and configuration files of the Web3 blog system. Explain how version control enables tracking changes, branching, merging, and maintaining a history of configurations.

Configuration Management Plan: Describe the configuration management plan, which outlines the overall strategy and approach for managing configurations within the Web3 blog system project. This plan may include details on naming conventions, file structure, release management, and deployment procedures.

Configuration Documentation: Highlight the significance of documenting configurations to aid in system maintenance, troubleshooting, and future enhancements. Discuss the documentation standards and formats used for capturing configuration details, including configuration diagrams, data dictionaries, and system dependencies.

Configuration Security: Discuss the measures taken to ensure the security of configurations, including access controls, encryption, and secure storage of sensitive configuration information. Emphasize the importance of protecting configurations from unauthorized access or modifications.

9.1 Installation

Operating System: Choose a compatible operating system such as Windows, macOS, or Linux.

Node.js: Install Node.js, a JavaScript runtime, which will be used to run the backend server and build the frontend application.

Package Manager: Install a package manager like npm (Node Package Manager) or Yarn to manage dependencies and install required libraries.

Text Editor or Integrated Development Environment (IDE): Choose a text editor or IDE of your preference for editing code files.

Blockchain Network: Decide on the blockchain network you will be using, such as Ethereum. Set up a local development blockchain or connect to a testnet to interact with smart contracts.

Smart Contract Development Tools: Install development tools such as Solidity compiler (solc), Truffle, or Hardhat to compile and deploy smart contracts.

Decentralized Storage: Choose a decentralized storage solution like IPFS (InterPlanetary File System) for storing blog content and media files.

Backend Dependencies: Install backend dependencies using the package manager. These may include web frameworks like database libraries, and libraries for interacting with the blockchain network.

Frontend Dependencies: Install frontend dependencies using the package manager. These may include frameworks like React.js, styling libraries like Bootstrap or Tailwind CSS, and Web3 libraries for interacting with the blockchain.

9.2 User Manual:

Login –

MetaMask is a popular wallet browser extension that enables users to interact with Web3 applications, including Web3 sites. Here's a description of the implementation and functionality of MetaMask wallet for a Web3 site:

Implementation:

- The Web3 site integrates the MetaMask wallet by utilizing the MetaMask browser extension or injecting the MetaMask library into the site's JavaScript code.

- The site typically includes a login or connect button that triggers the MetaMask wallet prompt when clicked.

- The site's backend communicates with the MetaMask wallet through the Web3.js library or similar Ethereum-compatible libraries to interact with the Ethereum blockchain.

Used Libraries -

1) Ethers – For the integration of front end and back end
2) Web3Modal – To connect with Wallet using metamask button
3) @walletconnect/web3-provider - To connect with the Wallet using QR code

```
async function connect() {
  try {
    const web3Modal = await getWeb3Modal()
    const connection = await web3Modal.connect()
    const provider = new ethers.providers.Web3Provider(connection)
    const accounts = await provider.listAccounts()
    setAccount(accounts[0])
  } catch (err) {
    console.log('error:', err)
  }
}
```

Fig. 9.2.1 Login Function

Screenshots:

First visit the localhost, user will see this home page:

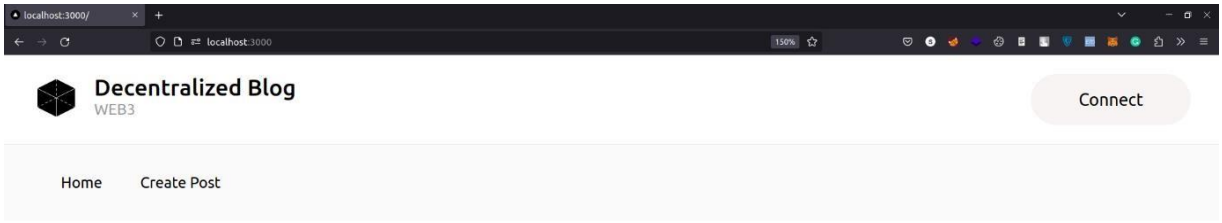

Fig. 9.2.2 Home Page

After clicking the Connect button user will see these options. User select the preferable option to login into the system:

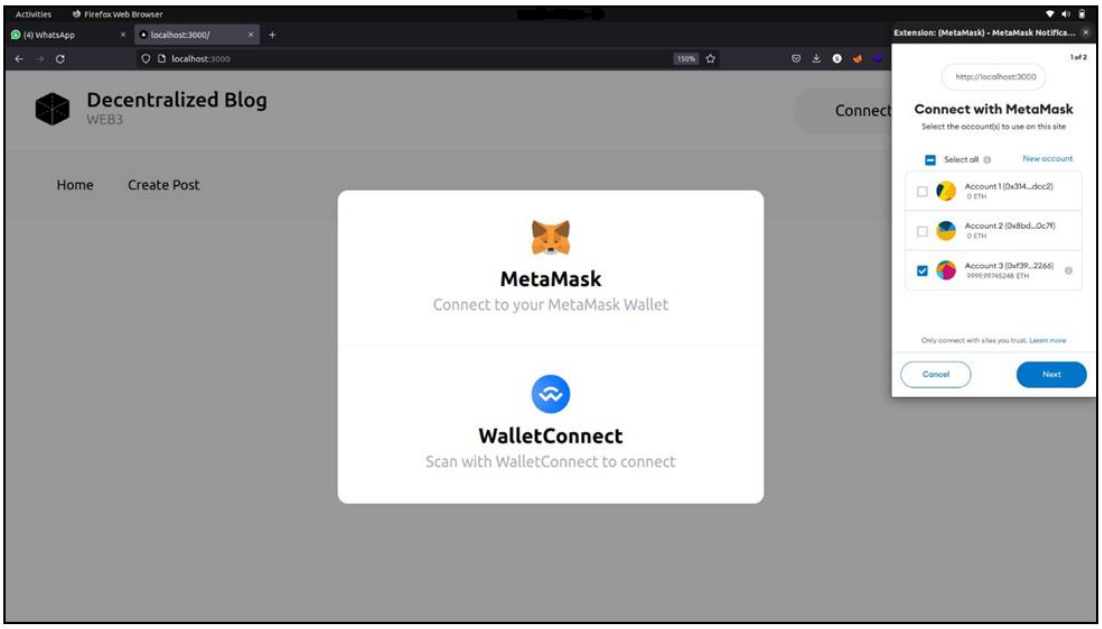

Fig. 9.2.3 After Clicking Connect Button

After successfully logged in into system, user will see their signature address and the create post option will be available:

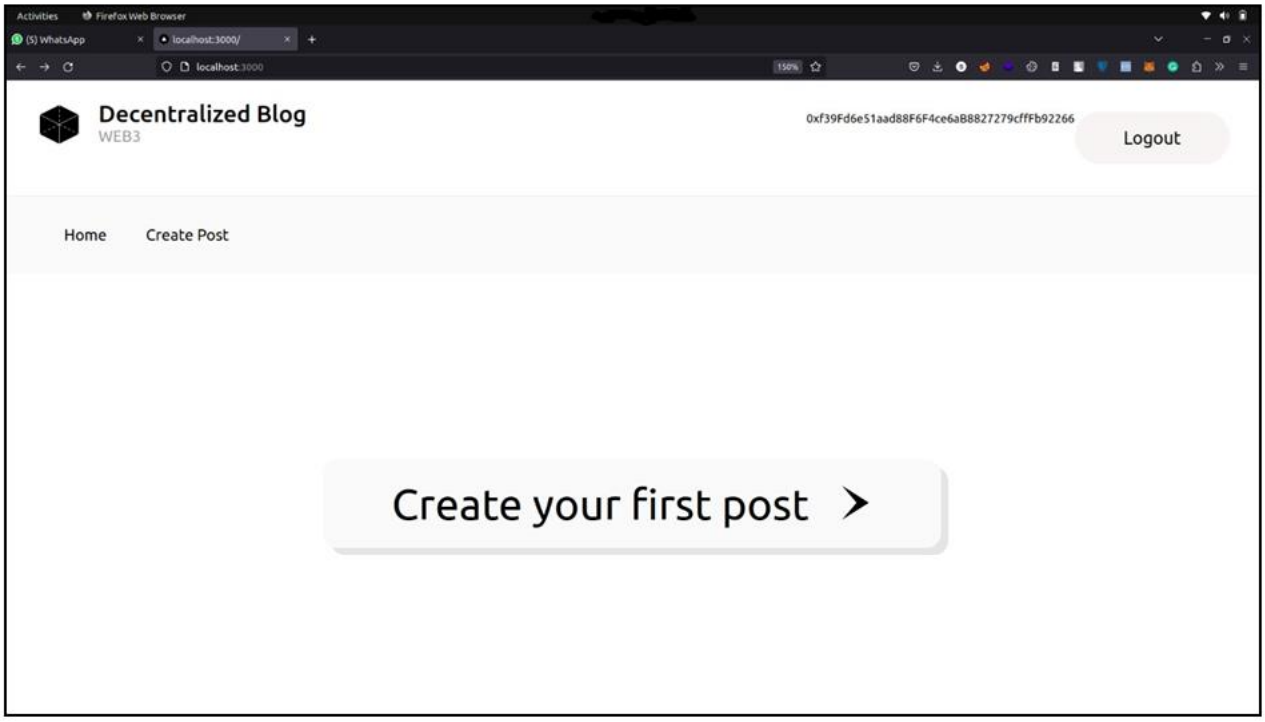

Fig. 9.2.4 After Login

Create Blog –

Implementation:

1. User Authentication: The Web3 blog site integrates the MetaMask wallet for user authentication. Users are required to connect their MetaMask wallet to the site to prove ownership of their Ethereum account.

2. Smart Contract Integration: The Web3 blog site utilizes a smart contract deployed on the Ethereum blockchain to manage blog posts. The contract contains functions to create, update, and retrieve blog posts.

3. User Interface: The site includes a user interface where users can input the necessary details for creating a blog post, such as the title, content, tags, and optional features like images and categories. The UI also displays the estimated gas fee for the transaction.

Functionality:

1. Wallet Authentication: Upon accessing the blog post creation page, users are prompted to connect their MetaMask wallet. This authentication process ensures that only authorized users can create blog posts.

2. Input Validation: The Web3 blog site validates user input to ensure that required fields, such as the title and content, are not empty. It may also enforce character limits or perform additional validation checks for the content.

3. Transaction Preparation: Once the user fills in the necessary details, the Web3 blog site prepares a transaction to interact with the smart contract. The transaction includes the function call to create a blog post, along with the provided input values.

4. Gas Fee Calculation: The MetaMask wallet estimates the gas fee required for executing the transaction. The user is presented with the estimated gas fee, giving them the option to adjust the gas price or confirm the transaction with the suggested fee.

5. Transaction Signing and Submission: After the user confirms the transaction, the MetaMask wallet prompts them to review and sign the transaction using their private key. Once the user signs the transaction, MetaMask broadcasts it to the Ethereum network for execution.

Libraries Used -

1) Ethers – to interact with smart contract.
2) Ipfs-http-client – to create IPFS clinet.
3) react-simplemde-editor – to create interface for writing the blog.

We are using INFURA provider to interact with IPFS database -

Set up an Infura account: Visit the Infura website (https://infura.io) and create an account. After signing up, you'll receive a Project ID and Project Secret, which you'll need to interact with Infura's IPFS API.

Connect to the Infura IPFS API: In your Web3 blog system project, establish a connection to the Infura IPFS API. This can be done using a Web3 library such as web3.js or ethers.js. Create an instance of the library and provide the Infura endpoint as the connection URL.

Upload the blog content to IPFS: Convert your blog content (text, images, etc.) into an IPFS-compatible format. This involves packaging the content into a directory or using an IPFS library to create a suitable IPFS object. Then, use the web3.js library to add the content to IPFS via the Infura IPFS API.

```
const projectId = 'XXXX'
const projectSecret = 'XXXX'
const auth =
  'Basic ' + Buffer.from(projectId + ':' + projectSecret).toString('base64')
const client = create({
  host: 'ipfs.infura.io',
  port: 5001,
  protocol: 'https',
  headers: {
    authorization: auth
  }
})
```

Fig. 9.2.5 Infura Client

Screenshots:

User will see this interface to write a blog. Add Title, Content and Cover Image.

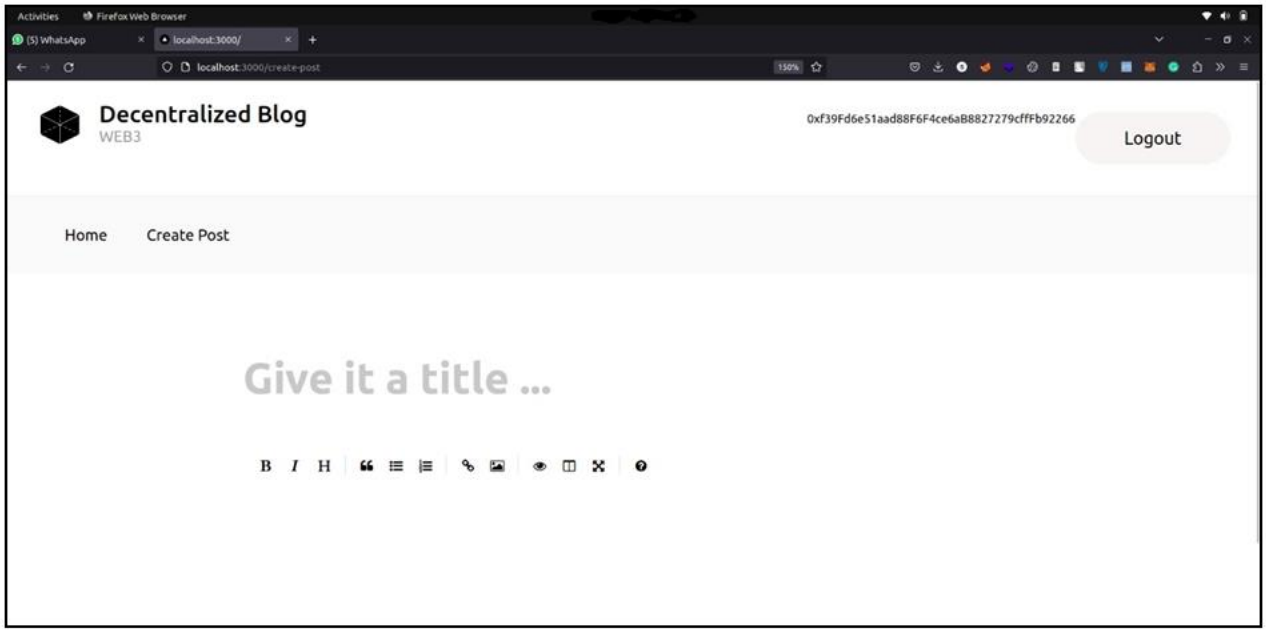

Fig. 9.2.6 After Clicking Create Post Option

After adding the content, click on the publish button to post the blog. Metamask extention will popped-up to confirm the transaction:

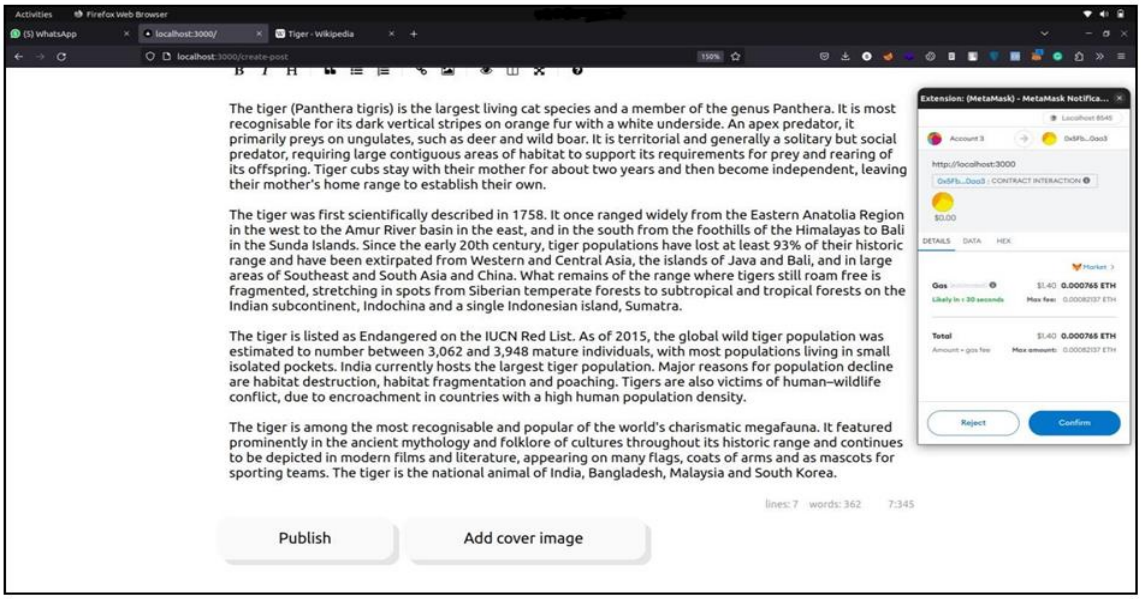

Fig. 9.2.7 After Clicking Publish Button

Read Blog –

Implementation:

1. Smart Contract Integration: The Web3 blog site incorporates a smart contract deployed on the Ethereum blockchain. The smart contract includes functions to store and retrieve the hash value of blog posts.

2. IPFS Integration: The Web3 blog site integrates with IPFS (InterPlanetary File System), a decentralized file storage system. The hash value obtained from the smart contract is used to retrieve the actual content of the blog post from IPFS.

3. User Interface: The site's user interface allows users to browse and select blog posts they want to read. The UI displays relevant information such as the title, author, and summary of each blog post.

Functionality:

1. Fetching Hash Value: When a user selects a blog post to read, the Web3 blog site calls a function on the smart contract to retrieve the hash value associated with that specific blog post. The hash value serves as a reference to the content stored on IPFS.

2. IPFS Retrieval: Using the obtained hash value, the Web3 blog site sends a request to IPFS to retrieve the content associated with that particular hash value. IPFS locates and retrieves the blog post content, which may include the title, author, date, and the main text of the post.

3. Displaying Blog Post: Once the content is fetched from IPFS, the Web3 blog site displays the blog post in the user interface, presenting it in a readable format. The title, author, and other relevant information are displayed along with the main text of the post.

4. Immutable and Verifiable Content: Since the blog post content is stored on IPFS and referenced by its hash value in the smart contract, the content becomes immutable. Users can verify the integrity of the content by comparing the retrieved hash value with the one stored on the smart contract.

Libraries used -

1) Ethers – to interact with smart contract.
2) Ipfs-http-client – to create IPFS clinet.

We are using INFURA provider to fetch the blog from IPFS

Fetch the blog content: To fetch a blog from IPFS, you'll need to know the IPFS hash of the content. In your Web3 blog system, you can store the IPFS hash associated with each blog post. Using the IPFS hash, you can request the content from the IPFS network through the Infura IPFS API.

```
export async function getStaticProps({ params }) {

  const { id } = params
  const ipfsUrl = `${ipfsURI}/${id}`
  const response = await fetch(ipfsUrl)
  const data = await response.json()
  if(data.coverImage) {
    let coverImage = `${ipfsURI}/${data.coverImage}`
    data.coverImage = coverImage
  }

  return {
    props: {
      post: data
    },
  }
}
```

Fig. 9.2.8 Fetch Post Function

Screenshots:

After posting the blog, user will the blog posts in the feed:

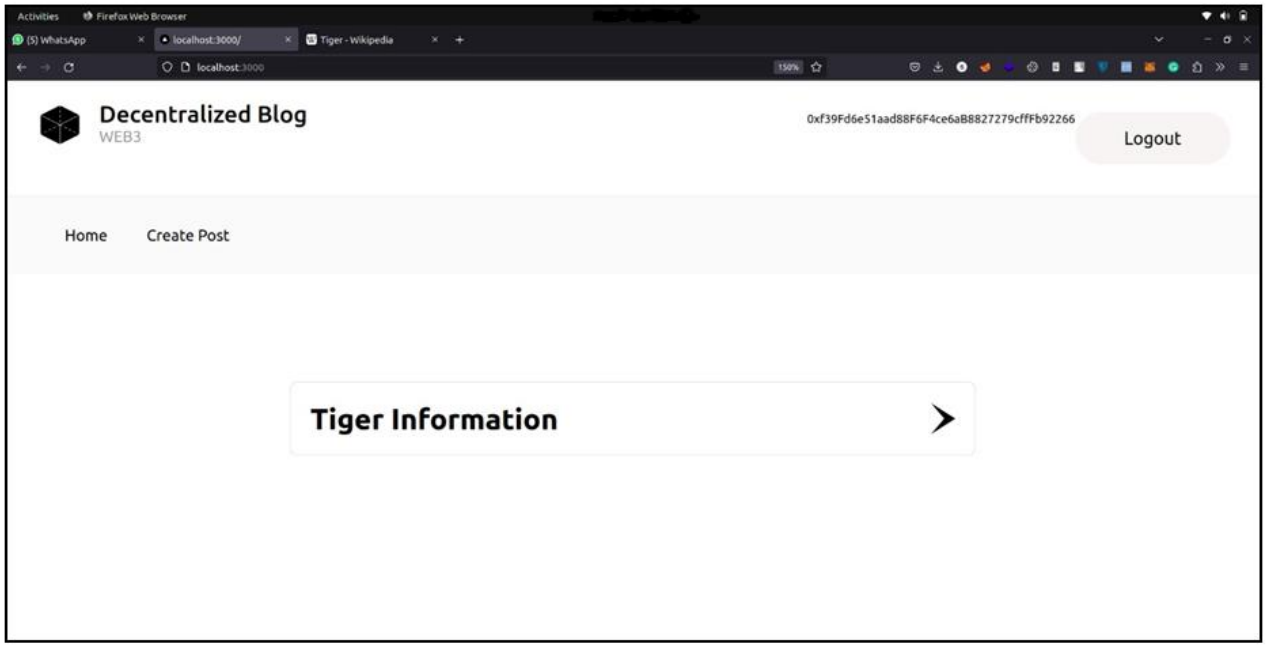

Fig. 9.2.9 After Publishing Blog

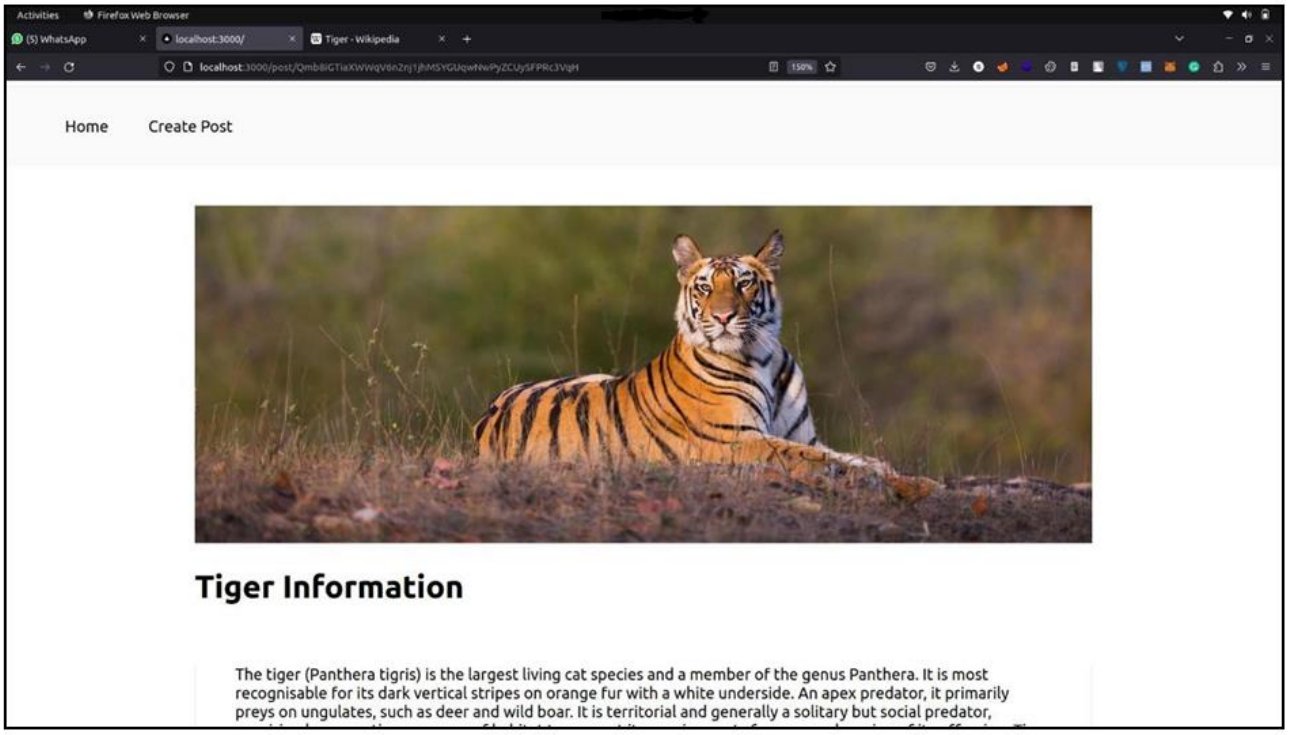

Fig. 9.2.10 Blog View

5.2 Logout –

Functionality:

1. Session Clearance: When a user clicks the logout button or selects the logout option, the Web3 blog website clears the user's session data, including any stored authentication tokens or session identifiers.

2. Access Restriction: Upon logout, the Web3 blog website restricts access to certain functionalities and features that require authentication. For example, creating new blog posts or accessing user-specific settings may no longer be available until the user logs back in.

3. Wallet Disconnect : If the Web3 blog website is integrated with a wallet provider, such as MetaMask, an optional step may involve disconnecting the wallet from the site. This ensures that the user's wallet connection is terminated, preventing unintended interactions with the blockchain.

```
async function logout() {
  setAccount(null);
  setIsLoggedIn(false);
}
```

Fig. 9.2.11 Logout Function

Screenshot –

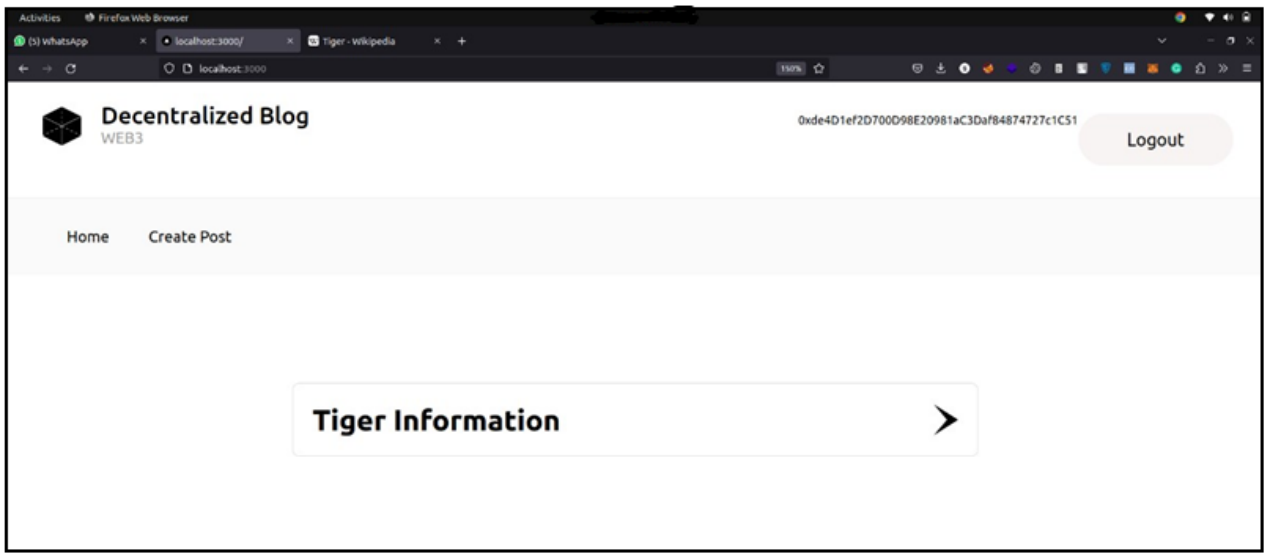

Fig. 9.2.12 Logout Button Visibility

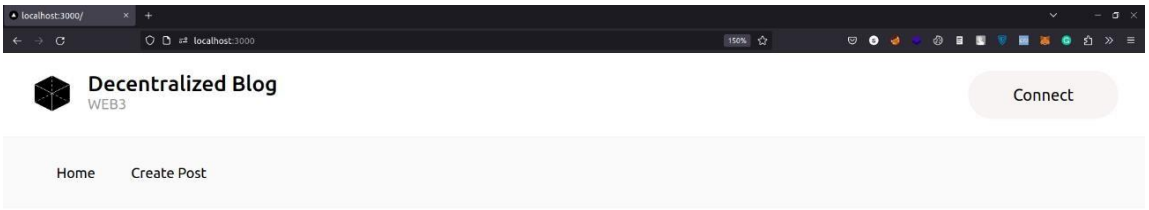

Fig. 9.2.13 After Logout

Chapter 10
Conclusion and Future Scope

11.1 Conclusion:

In conclusion, the Web3 Blog System Project has been successfully completed, resulting in the development of a decentralized blogging platform that leverages Web3 technologies. Throughout the project, we aimed to address the limitations of traditional centralized blogging systems by harnessing the power of blockchain, smart contracts, and decentralized storage.

The project objectives were effectively achieved through careful planning, implementation, and testing. The Web3 blog system offers several key advantages over traditional platforms, including enhanced data ownership, censorship resistance, and increased transparency.

The project successfully achieved its objectives by developing a Web3 blog system that leverages the decentralized nature of blockchain networks, ensuring data integrity, censorship resistance, and user ownership of content. By integrating with a specific blockchain network, such as Ethereum, and utilizing decentralized storage solutions like IPFS, the system provides a resilient and transparent platform for publishing and sharing blog posts.

The report discussed the project scope, clearly defining the in-scope and out-of-scope aspects. We focused on the core functionalities of the blog system, including user registration, content creation, publishing, content discovery, and basic social features. While certain advanced features and scalability considerations were beyond the project boundaries, the system serves as a solid foundation for future enhancements.

By utilizing blockchain technology, the system ensures the integrity and immutability of blog posts, providing users with the confidence that their content cannot be tampered with or modified without their knowledge. The integration of smart contracts facilitates transparent interactions between users, enabling functionalities such as user authentication, content creation, and interaction tracking.

Decentralized storage solutions, such as IPFS, enable the system to store media files associated with blog posts in a distributed manner, ensuring redundancy, availability, and resistance to single-point failures.

Throughout the project, risk management strategies were employed to identify potential challenges and mitigate them proactively. Contingency plans were developed to address high-risk events, ensuring project progress and minimizing disruptions.

11.2 Future Scop:

The Web3 Blog System Project has laid a solid foundation for a decentralized and secure blogging platform. As the technology landscape continues to evolve, there are several avenues for future expansion and enhancement of the Web3 Blog System. Here are some potential areas of future scope:

Enhanced User Experience:

Focus on improving the user interface and user experience of the Web3 blog system to attract more users and encourage engagement. Incorporate modern design principles, intuitive navigation, and responsive layouts to ensure a seamless and enjoyable user experience across different devices.

Monetization and Tokenization:

Explore ways to integrate cryptocurrency and tokenization features into the Web3 blog system, allowing content creators to monetize their work and receive direct payments from users. Implement mechanisms for micropayments, donations, and subscription models to incentivize quality content creation and support the blogging community.

Social Features and Interactions:

Introduce social features such as user profiles, comments, likes, and sharing capabilities to foster community engagement and interaction among users. Implement features that allow users to follow their favorite bloggers, receive notifications, and discover trending or recommended content.

Content Discovery and Search:

Develop advanced search functionality to enable users to easily discover relevant blog posts based on keywords, categories, tags, or personalized recommendations.

Implement content filtering, sorting, and ranking algorithms to enhance the content discovery experience for users.

Collaboration and Co-authoring:

Introduce collaboration features that allow multiple authors to work together on a single blog post, enabling co-authoring, version control, and attribution mechanisms. Implement real-time editing and commenting capabilities to facilitate collaboration among bloggers and content contributors.

Mobile Application Development:

Extend the Web3 blog system by developing dedicated mobile applications for iOS and Android platforms, offering users the convenience of accessing and contributing to the platform on their mobile devices.

12.1 References:

8. Research Paper: A Blockchain based Autonomous Decentralized Online Social Network | IEEE Conference Publication | IEEE Xplore

9. Infura Doc: https://docs.infura.io/infura/

10. IPFS Doc: https://docs.ipfs.tech/concepts/what-is-ipfs/

11. React Doc: https://legacy.reactjs.org/docs/getting-started.html

12. Solidity Doc: https://docs.soliditylang.org/en/v0.8.20/

13. Web3 Doc: https://ethereum.org

14. Ethereum Doc: https://learnweb3.io