Software Requirements Specification
for

Decentralized Web Hosting

Members:

Aditya Ingle 112103050
Vedang Khedekar 112103070
Anagha Mahure 112103082

Table of Contents

1. Introduction

1.1 Purpose

The project focuses on creating a decentralized web hosting system using IPFS and Ethereum blockchain

Traditional hosting methods face issues like downtime and vulnerability due to centralization in a single point of control

To address centralization issues, the project employes IPFS for decentralized file storage and Ethereum blockchain for immutability and secure transactions

1.2 Intended Audience and Reading Suggestions

This document is intended for Blockchain Enthusiasts, Web developers,system administrator, Entrepreneurs,startups,students, researcher and stakeholders involved in the development and deployment. It provides a comprehensive guide to understanding the project goals, functionalities, and constraints, fostering collaboration and ensuring a unified vision across all stakeholders

1.3 Product Scope

Decentralized Storage: Utilizing blockchain for distributed storage of website content, reducing reliance on traditional servers.

Blockchain-based Identity: Implementing decentralized identity solutions to ensure secure and verifiable ownership of websites.

Smart Contracts for Hosting Agreements: Using smart contracts to automate hosting agreements, ensuring transparent and enforceable terms.

Content Delivery Network (CDN) Integration: Incorporating CDN capabilities to enhance website performance and reduce latency.

User-Friendly Interfaces: Developing intuitive interfaces for users to easily manage and interact with their decentralized hosting services.

2. Overall Description

2.1 Product Perspective

Create a user-friendly decentralized web hosting platform using blockchain for seamless adoption, prioritizing scalability, reliability, security. Continuously iterate to enhance the platform's features and address evolving user needs.

2.2 Product Functions

1. Decentralized File Storage:
   - Enable users to store and retrieve files in a decentralized manner, leveraging technologies like IPFS for distributed and resilient file storage.

2. Blockchain-based Authentication:
- Implement secure user authentication and access control using blockchain-based identity management, ensuring a trustless and tamper-resistant authentication system.

3. Content Addressing
- Utilize content addressing mechanisms, such as IPFS content addressing, to uniquely identify and retrieve content based on its cryptographic hash, ensuring integrity and authenticity.

4 Smart Contracts for Agreements:
- Implement smart contracts to automate and enforce agreements between users and storage providers, defining terms such as storage duration, pricing, and conditions.

7. User-Friendly Interfaces:
- Develop user-friendly interfaces, such as web portals or applications, to simplify interactions with the decentralized hosting platform, making it accessible to a broader audience.

9. Decentralized Domain Name System (DNS):
- Explore integrating or supporting decentralized DNS solutions to enable users to access content through hashable domain names on the decentralized web.

2.3 User Classes and Characteristics

Developers:
Characteristics: Web development expertise, blockchain interest.
Needs: Robust APIs, developer-friendly features.

Website Owners:
Characteristics: Diverse, from bloggers to businesses.
Needs: User-friendly interface, reliability, compatibility.

SMEs:
Characteristics: Small to medium-sized businesses.
Needs: Cost-effective, scalable hosting.

2.4 Operating Environment

The application will operate in a web-based environment
accessible via standard web browsers on desktop and mobile devices.

Operating Environment:
- Blockchain Infrastructure: Uses blockchain protocols.
- Security: Implements encryption, decentralized authentication.
- Community Ecosystem: Engages developers, node operators, and users.

2.5 Design and Implementation Constraints

1. Blockchain Scalability:
   - Constraint: Potential slowdowns in blockchain networks.

3. Network Latency:
   - Constraint: Increased latency in retrieving content from decentralized networks.

4. Regulatory Compliance:
   - Constraint: Navigating evolving legal frameworks for blockchain and data storage.

5. Security Risks:
   - Constraint: Smart contract vulnerabilities and data breaches.

3. External Interface Requirements

3.1 User Interfaces

For a decentralized web hosting project using blockchain, the user interface should include:

1. Registration/Login:
   - Allow users to create accounts securely and log in using blockchain-based authentication.

2. File Upload/Management:
   - Provide a user-friendly interface for uploading, managing, and organizing files within the decentralized storage system.

3. Content Preview:(Maybe)
   - Enable users to preview and verify the content they have stored, ensuring accuracy.

4. Smart Contract Interaction:
   - Integrate an interface for users to interact with smart contracts, setting terms for storage agreements with providers.

5. Token Wallet/Management:
   - Include a wallet interface for users to manage blockchain tokens earned or spent within the platform.

9. Notifications:
   - Implement a notification system to update users on storage agreements, transactions, and important platform updates.

3.2 Software Interfaces

Software Interfaces for Decentralized Web Hosting:

1. Blockchain Integration (Ethereum):
   - Node.js and Python: Interfaces for storing and managing IPFS hash addresses.

2. Smart Contract Development:
   - VS Code: Deploying contracts to record IPFS hashes for data immutability.

3. Verification System:
   - VS Code: Enables users to verify web content ownership via recorded hash codes.

4. Decentralized Storage Interaction:
   - Node.js: Interfaces for efficient storage/retrieval of content on IPFS.
5. Next JS
6. IPFS
7. Hardhat
8. Infura

3.3 Communications Interfaces

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

4. System Features

4.1 System Feature 1

The project focuses on creating a decentralized web hosting system using IPFS and Ethereum blockchain

Traditional hosting methods face issues like downtime and vulnerability due to centralizatiion in a single point of control

To address centralization issues, the project employes IPFS for decentralized file storage and Ethereum blockchain for immutability and secure transactions

Smart contracts in this project automate user registration, website deployment, and approval processes, ensuring data integity and linking IPFS-based website addresses to Ethereum blockchain for decentralized web hosting

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Performance Requirements for Decentralized Web Hosting:

- Scalability: Efficiently handle a growing user base.
- Throughput: High capacity for concurrent data requests and smart contract interactions.
- Latency: Minimize delays in content delivery and smart contract executions.
- Reliability: Ensure consistent availability of hosted content.
- Consensus Efficiency: Optimize blockchain consensus mechanisms.
- Node Performance: Specify minimum requirements for optimal node operation.
- Smart Contract Execution Time: Define acceptable processing times for smart contracts.
- Storage Efficiency: Optimize space utilization and retrieval times.
- Transaction Confirmation Speed: Specify desired transaction confirmation speed.
- Caching Mechanisms: Implement effective caching for faster data retrieval.
- Interoperability: Enable seamless integration with traditional hosting solutions.
- Resource Allocation Flexibility: Allow users to easily scale storage, bandwidth, and resources.
- Energy Efficiency: Strive for eco-friendly operations in decentralized hosting.
- CDN Integration Efficiency: Seamlessly integrate CDNs for optimized content delivery.

5.2 Security Requirements

- Load Balancing: Distribute requests evenly to prevent bottlenecks.
- Security Performance: Ensure robust security measures without compromising speed.
- Token Transaction Speed: Define desired speed for token transactions.
- Monitoring and Analytics: Implement tools for performance tracking and issue identification.
- Upgradeability: Support seamless system upgrades without significant downtime

Testing and Quality Assurance:

- **Unit Testing:** Conduct thorough unit testing of individual components to ensure functionality and identify potential bugs or issues.
- **Integration Testing:** Perform integration testing to validate interactions between system components and external dependencies.

Documentation:

- **User Guide:** Provide comprehensive documentation, including user guides and tutorials, to help users navigate the application effectively.
- **Developer Documentation:** Document the system architecture, APIs, and implementation details for developers to understand and extend the system.