**INDEX**

# INTRODUCTION

* 1. Project Overview
  2. Purpose

# LITERATURE SURVEY

* 1. Existing problem
  2. References
  3. Problem Statement Definition

# IDEATION & PROPOSED SOLUTION

* 1. Empathy Map Canvas
  2. Ideation & Brainstorming

# REQUIREMENT ANALYSIS

* 1. Functional requirement
  2. Non-Functional requirements

# PROJECT DESIGN

* 1. Data Flow Diagrams & User Stories
  2. Solution Architecture

# PROJECT PLANNING & SCHEDULING

* 1. Technical Architecture
  2. Sprint Planning & Estimation
  3. Sprint Delivery Schedule

# CODING & SOLUTIONING (Explain the features added in the project along with code)

* 1. Feature 1
  2. Feature 2
  3. Database Schema (if Applicable)

# PERFORMANCE TESTING

* 1. Performace Metrics

# RESULTS

* 1. Output Screenshots

1. **ADVANTAGES & DISADVANTAGES**
2. **CONCLUSION**
3. **FUTURE SCOPE**
4. **APPENDIX**

Source Code

GitHub Link  
Demo Video link

**NAAN MUDHALVAN – GUIDED PROJECT** 

**DOCUMENTATION**

|  |  |
| --- | --- |
| DATE | 31 OCTOBER 2023 |
| TEAM MEMBERS WITH NM ID | 1. HEMAH P   NM ID: 26C8ACB2D68B750B23 948062A9FDBD7B   1. JAYASHREE S   NM ID: ED018CE4032A071BB3C45143295B8FD7   1. KARAN R   NM ID: 6DD5B3DC8A86F23A1814299AF568A66A   1. KUMARAGHAN S M   NM ID: D8C6A6C4026C2CFA0294CEA4BB99ED6C |
| PROJECT NAME | DIGITAL ASSET MANAGEMENT USING BLOCKCHAIN TECHNOLOGY |
| COLLEGE NAME | ALAGAPPA COLLEGE OF TECHNOLOGY, ANNA UNIVERSITY |

**1. INTRODUCTION:**

Digital asset management through Ethereum involves using the Ethereum blockchain to manage digital assets. Ethereum is a blockchain-based platform that enables developers to build decentralized applications (dApps) and smart contracts. Digital assets are generally anything that is created and stored digitally, is identifiable and discoverable, and has or provides value.Examples of digital assets include cryptocurrencies, non-fungible tokens (NFTs), and tokenized assets

To manage digital assets through Ethereum, users can create smart contracts that define the rules and conditions for the asset. These smart contracts are then stored on the Ethereum blockchain, which ensures that they are immutable and transparent. Users can then interact with these smart contracts to transfer ownership of the digital asset, or to perform other actions.

**1.1 Project Overview:**

The project aims to develop a digital asset management system using Ethereum blockchain technology. The system will allow users to create, store, and transfer digital assets securely and transparently. The project will involve the following key components: Ethereum Integration, Smart Contract Development, Asset Creation and Storage, Asset Transfer and Ownership and User Interface

**1.2 Purpose of the project:**

The project aims to utilize Ethereum blockchain for secure and efficient digital asset management. Key objectives include enhancing security through blockchain's immutability, ensuring transparency with smart contracts, exploring new business models like asset tokenization, enabling easy asset exchange, and educating users on Ethereum and digital asset management for informed decision-making

**2. LITERATURE SURVEY:**

**2.1 Existing Problems:**

The business problem for Digital Asset Management on the Ethereum Blockchain revolves around the challenges associated with effectively managing, securing, and preserving the integrity of digital assets.In summary, the business problem for Digital Asset Management on the Ethereum Blockchain centers on the need to address these challenges effectively. Implementing blockchain-based solutions can provide transparency, security, and a tamper-proof record of digital asset ownership and transactions

Issue: Digital assets, such as images, videos, documents, and creative works, are susceptible to piracy and unauthorized duplication.

Impact: Creators and content owners suffer revenue losses, loss of control over their intellectual property, and potential damage to their brand reputation

1. "A Blockchain-Based Framework for Secure Digital Asset Management"[1]: This paper proposes a framework for secure digital asset management using Ethereum's smart contracts. The framework includes a multi-authority attribute-based access control scheme that ensures secure and transparent transactions of digital assets.

2. "Blockchains and Digital Assets"[2]: This academic paper provides an overview of how blockchains are being used to support the exchange of digital assets. The paper explores the potential of tokenization for real-world assets, the use of smart contracts for efficient and auditable record-keeping, and the creation of innovative financial instruments using blockchain technology.

3. "About Ethereum"[3]: This paper provides a primer on the basics of Ethereum and its potential applications. The paper explores the Ethereum ecosystem, including its smart contract functionality, decentralized nature, and potential for creating new business models and revenue streams.

4. "Blockchain Meets Metaverse and Digital Asset Management: A Comprehensive Survey"[4]: This paper provides a comprehensive survey of the use of blockchain technology in digital asset management, with a focus on use cases related to mobile edge computing and communication networks.

5. "Understanding Blockchain and Digital Assets"[5]: This paper provides a general overview of blockchain technology and its potential applications in digital asset management. The paper explores the different types of digital assets, including cryptocurrencies, tokens, and NFTs, and their potential as an investable asset class.

**2.2 Reference:**

[1] https://www.techrxiv.org/articles/preprint/A\_Blockchain-Based\_Framework\_for\_Secure\_Digital\_Asset\_Management/23796204

[2] https://www.eublockchainforum.eu/sites/default/files/research-paper/blockchains\_and\_digital\_assets\_june\_version.pdf

[3] https://www.fidelitydigitalassets.com/research-and-insights/about-ethereum

[4] https://ieeexplore.ieee.org/iel7/6287639/10005208/10068493.pdf

[5] https://www.manning-napier.com/insights/understanding-blockchain-and-digital-assets

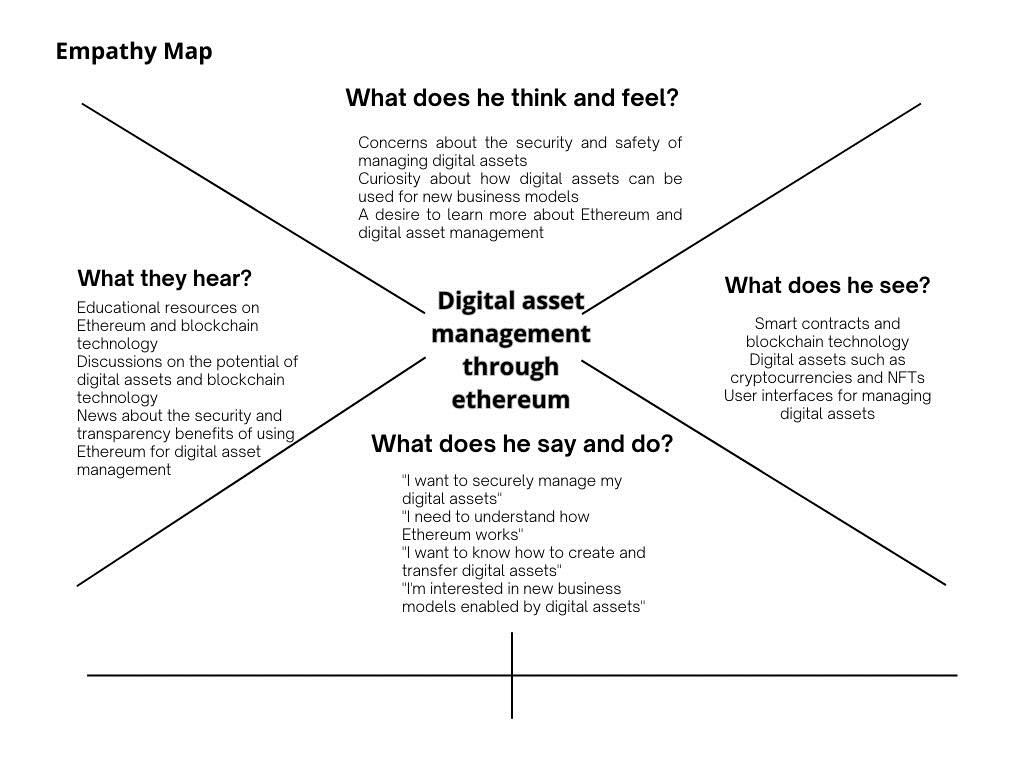
[6] https://www.researchgate.net/publication/372764946\_A\_Blockchain-Based\_Framework\_for\_Secure\_Digital\_Asset\_Management

**2.3 Problem Statement Definition:**

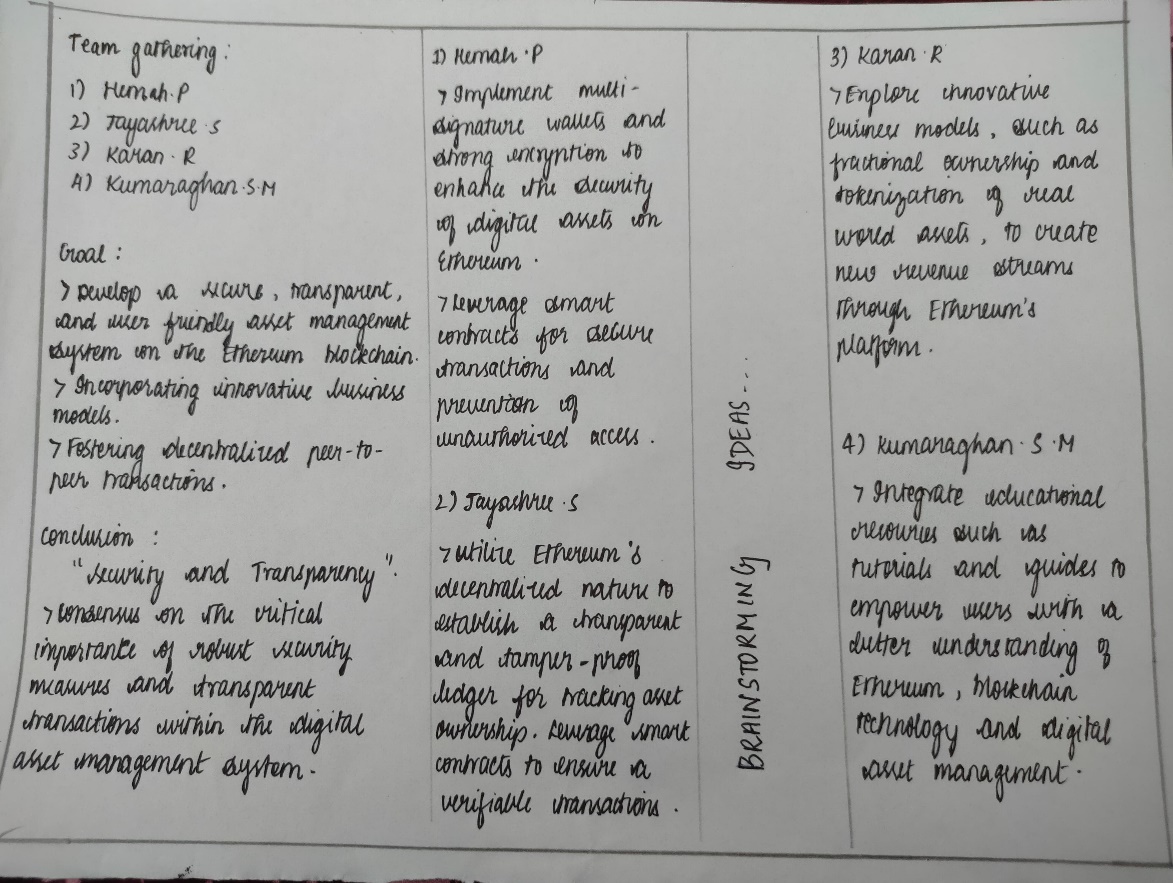
In the modern digital landscape, traditional digital asset management systems face challenges related to centralized control, security vulnerabilities, and a lack of transparent ownership tracking. To address these issues, there is a need for a decentralized and secure solution. The proposed Digital Asset Management system using blockchain aims to provide transparency, integrity, and efficient management of digital assets

**3. IDEATION & PROPOSED SOLUTION:**

**3.1 Empathy Map Canvas:**



**.2Ideation & Brainstorming:**



**4. REQUIREMENT ANALYSIS :**

**4.1 FUNCTIONAL REQUIREMENT:**

Drive link of the source code file: <https://drive.google.com/file/d/1EcRTLuhwfzK1gSi7pFBnQ-YFrKhFZ8Gh/view>

|  |  |  |
| --- | --- | --- |
| **Fr No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | Decentralized Database | Utilize blockchain for a secure, transparent, and tamperresistant database of library records. Explore blockchain platforms like Ethereum,  Hyperledger, or others for their capabilities in handling decentralized databases.  Consider data partitioning and distribution strategies for efficient storage and retrieval. |
| FR-2 | Smart Contracts | Implement smart contracts for automated processes like lending, returns, and overdue fines. Define the logic for sma contracts, specifying conditions for book borrowing, return deadlines, and penalty calculations.  Ensure smart contracts are audited for security and efficiency. |
| FR-3 | User Identity Management | Ensure a robust system for managing user identities, borrowing history, and preferences securely on the blockchain. |
| FR-4 | Immutable Record Keeping | Leverage blockchain's immutability to maintain an unalterable history of transactions and library assets. Implement a robust user authentication system with cryptographic keys or biometrics.  Research decentralized identity solutions for managing user credentials securely. |
| FR-5 | Data encryption | Implement robust encryption mechanisms to protect sensitive user data and maintain privacy. |
| FR-6 | Privacy and Permissions | Implement granular access controls to ensure data privacy and limit access based on user roles within the library eco- system.  Implement encryption techniques to protect sensitive user data.  Define roles and permissions for different user types (librarians, administrators, users) to control access. |
| FR-7 | Tokenization | Design a tokenomics model, specifying how tokens are earned, spent, and exchanged within the library eco- system.  Consider integrating a wallet system for users to manage their tokens. |
| FR-8 | Integration with External Systems: | Research API compatibility with existing library management systems.  Ensure seamless data flow between your blockchain system and external databases. |
| FR-9 | Audit Trails | Implement a transparent and accessible audit trail accessible to authorized parties.  Consider visualizations or reporting tools for better interpretability of audit data. |
| FR-10 | Decentralized Consensus Mechanism | Choose an appropriate consensus mechanism (e.g., Proof of Work or Proof of Stake) to secure and validate transac- tions within the library network |

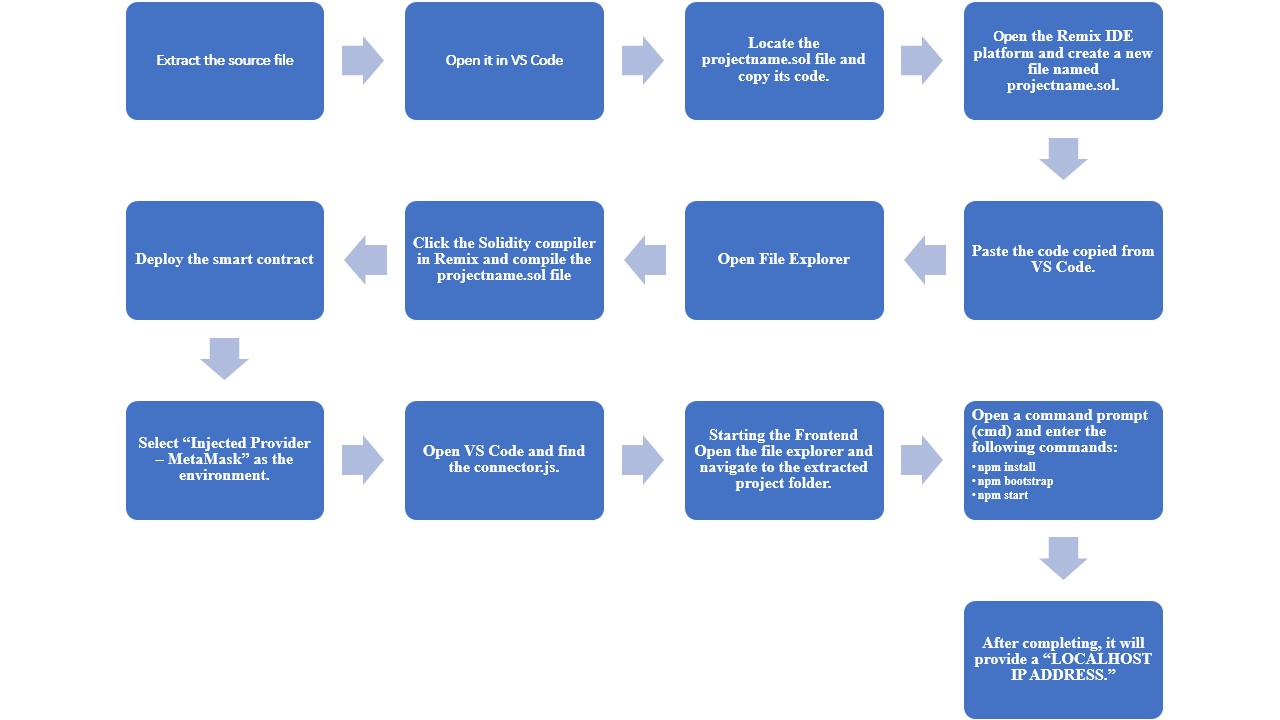
**4.2 NON-FUNCTIONAL REQUIREMENT:**

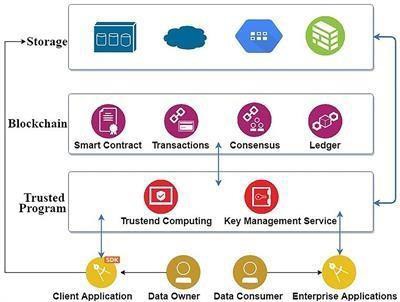
|  |  |  |  |
| --- | --- | --- | --- |
| **NFR**  **no.** | **Non functional requirements** | **Sub Requirement (Story / Sub-Task)** | |
| NFR-1 | Performance | The system should handle a large number of simultaneous transactions efficiently.  Response time for user interactions (e.g., search,check- out) should be within acceptable limits. | |
| NFR-2 | Reliability | The system should have high availability to ensure access to library services at all times.  Mean Time Between Failures (MTBF) should meet or exceed specified standards. | |
| NFR-3 | Security | Implement robust encryption algorithms to secure data transmission and storage.  Regular security audits and updates to address vulnerabilities. | |
| NFR-4 | Usability | The user interface should be intuitive and user-friendly to accommodate users of varying technical expertise.  Accessibility standards should be followed to ensure inclusivity. | |
| NFR-5 | Interoperability | The system should be able to integrate seamlessly with other library systems or external databases.  Support standard data exchange formats. | |
| NFR-6 | Auditability | Maintain detailed logs of transactions and systemac- tivities for auditing purposes.  Ensure transparency in system operations. | |
| NRF-7 | Regulatory Compliance | Adhere to relevant data protection regulations and standards.  Ensure compliance with library and educational institution policies. |

**5. PROJECT DESIGN:**

**5.1 Data Flow Diagram & User Stories**

**5.1.1 Data Flow Diagram**



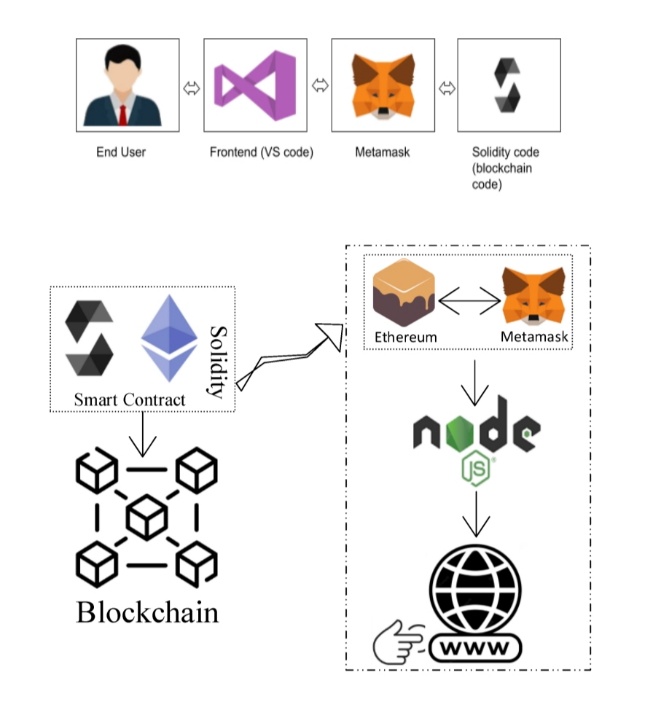


**5.1.2 USER STORIES:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Serial Number** | **User Story Number** | **User Story / Task** | **Acceptance Criteria** | **Priority** | **Release** |
| 1 | US-001 | Create project plan | The project plan should include tasks, timelines, and resource allocation. | High | Sprint 1 |
| 2 | US-002 | Set up blockchain environment | Ensure that the blockchain environment is functional and supports project needs. | High | Sprint 1 |
| 3 | US-003 | Define project scope | Clearly articulate the scope and requirements of the Digital Asset Management system. | High | Sprint 1 |
| 4 | US-004 | Create smart contracts for Asset handling | Develop smart contracts that facilitate digital asset management on the blockchain. | High | Sprint 2 |
| 5 | US-005 | Implement user registration and authentication | Users should be able to register and authenticate on the blockchain platform. | High | Sprint 2 |
| 6 | US-006 | Implement asset upload functionality | Allow users to upload digital assets to the blockchain platform. | High | Sprint 3 |
| 7 | US-007 | Ensure secure storage of digital assets | Implement a secure mechanism for storing digital assets on the blockchain. | High | Sprint 3 |
| 8 | US-008 | Define user roles and permissions | Admins should be able to define user roles and associated permissions. | High | Sprint 4 |
| 9 | US-009 | Implement differentiated access levels | Users should have different levels of access based on their roles. | High | Sprint 4 |
| 10 | US-010 | Create an intuitive user interface | Develop a user-friendly interface for interacting with digital assets. | High | Sprint 5 |
| 11 | US-011 | Perform thorough testing and bug fixes | Ensure the system is thoroughly tested for security and functionality. | High | Sprint 5 |

**5.2 SOLUTION ARCHITECTURE:**

The proposed solution architecture for the Digital Asset Management system integrates various technologies seamlessly within Visual Studio Code, offering a comprehensive development environment. Leveraging Hyperledger Fabric for a permissioned blockchain, smart contracts are developed using Solidity in Visual Studio Code, with Remix IDE providing a web-based interface and node.js connector for testing. The system incorporates IPFS for decentralized storage, with file explorer integration in Visual Studio Code facilitating efficient asset management. User authentication is secured through blockchain IDs and MetaMask integration. The user interface is designed using HTML, CSS, and JavaScript within Visual Studio Code, ensuring compatibility with the MetaMask Chrome extension. Security measures include end-to-end encryption and MetaMask for secure transactions. Scalability is addressed through sharding strategies and caching mechanisms integrated into Visual Studio Code. Real-time monitoring and an analytics dashboard enhance visibility into blockchain activity and system performance. This architecture thus creates a unified, secure, and user-friendly ecosystem for developing and deploying a robust Digital Asset Management system.

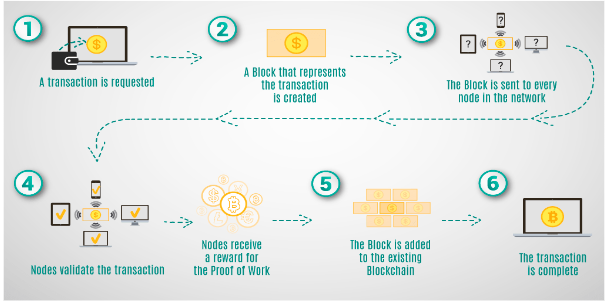


**6. PROJECT PLANNING & SCHEDULING:**

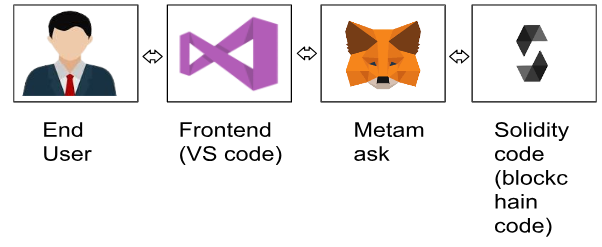
**6.1 Technical Architecture:**

This proposed smart contract leverages the Ethereum blockchain to create a secure and tamper-proof system for managing digital assets while preserving ownership, preventing piracy, and ensuring data integrity. It addresses the challenges associated with the duplication and unauthorized distribution of digital assets.By implementing this smart contract on the Ethereum blockchain, digital asset owners can effectively prevent unauthorized duplication and distribution, preserve ownership, maintain data integrity, and have the ability to update asset details as required. This solution has the potential to revolutionize digital asset management and protect creators' intellectual property rights in the digital age.

**Technical Architecture:**



**Technical Stack:**



**6.2 Sprint Planning & Estimation:**

In this agile sprint planning and estimation for the Digital Asset Management project using blockchain, we've organized the development process into five one-week sprints, each with distinct goals and deliverables. The initial sprint focuses on project initiation, setting up the blockchain environment, and defining the project scope. Subsequent sprints address critical functionalities, such as smart contract development, asset uploading, secure storage, access control, and user permissions. Each sprint comprises specific user stories, estimated tasks, and corresponding deliverables, promoting a clear and iterative development process. Sprint reviews and retrospectives at the end of each cycle ensure that the team can adapt, refine, and enhance the project based on feedback, evolving requirements, and emerging insights, fostering an agile and responsive development methodology.

**6.3 Sprint Delivery Schedule:**

Sprint 1: Initialization and Planning

Goal: Set up project infrastructure and define scope.

User Stories:

Create a project plan.

Set up blockchain environment.

Define project scope and requirements.

Estimation:

Task 1: 2 days

Task 2: 3 days

Task 3: 2 days

Deliverables:

Project plan

Initialized blockchain

Scope and requirements document

Sprint 2: Smart Contract Development

Goal: Develop smart contracts for asset management.

User Stories:

Create smart contracts for asset handling.

Implement user registration and authentication.

Estimation:

Task 1: 5 days

Task 2: 3 days

Deliverables:

Asset management smart contracts

User authentication

Sprint 3: Asset Upload and Storage

Goal: Enable uploading and secure storage of digital assets.

User Stories:

Implement asset upload functionality.

Ensure secure storage of digital assets.

Estimation:

Task 1: 4 days

Task 2: 4 days

Deliverables:

Asset upload feature

Secure asset storage

Sprint 4: Access Control and Permissions

Goal: Implement access control and user permissions.

User Stories:

Define user roles and permissions.

Implement differentiated access levels.

Estimation:

Task 1: 3 days

Task 2: 4 days

Deliverables:

User roles and permissions

Differentiated access levels

Sprint 5: User Interface and Testing

Goal: Develop a user-friendly interface and conduct testing.

User Stories:

Create an intuitive user interface.

Perform thorough testing and bug fixes.

Estimation:

Task 1: 4 days

Task 2: 3 days

Deliverables:

User-friendly interface

Completed testing and bug fixes

**7. CODING AND SOLUTIONING:.**

**CODING:**

const { ethers } = require("ethers");

const abi = [

{

"anonymous": false,

"inputs": [

{

"indexed": false,

"internalType": "string",

"name": "assetHash",

"type": "string"

},

{

"indexed": false,

"internalType": "address",

"name": "newOwner",

"type": "address"

}

],

"name": "AssetOwnershipTransferred",

"type": "event"

},

{

"anonymous": false,

"inputs": [

{

"indexed": false,

"internalType": "string",

"name": "assetHash",

"type": "string"

}

],

"name": "AssetPublished",

"type": "event"

},

{

"anonymous": false,

"inputs": [

{

"indexed": false,

"internalType": "address",

"name": "owner",

"type": "address"

},

{

"indexed": false,

"internalType": "string",

"name": "assetHash",

"type": "string"

}

],

"name": "AssetRegistered",

"type": "event"

},

{

"anonymous": false,

"inputs": [

{

"indexed": false,

"internalType": "string",

"name": "assetHash",

"type": "string"

}

],

"name": "AssetUnpublished",

"type": "event"

},

{

"inputs": [

{

"internalType": "string",

"name": "",

"type": "string"

}

],

"name": "digitalAssets",

"outputs": [

{

"internalType": "address",

"name": "owner",

"type": "address"

},

{

"internalType": "string",

"name": "assetHash",

"type": "string"

},

{

"internalType": "bool",

"name": "isPublished",

"type": "bool"

}

],

"stateMutability": "view",

"type": "function"

},

{

"inputs": [

{

"internalType": "string",

"name": "\_assetHash",

"type": "string"

}

],

"name": "publishAsset",

"outputs": [],

"stateMutability": "nonpayable",

"type": "function"

},

{

"inputs": [

{

"internalType": "string",

"name": "\_assetHash",

"type": "string"

}

],

"name": "registerAsset",

"outputs": [],

"stateMutability": "nonpayable",

"type": "function"

},

{

"inputs": [

{

"internalType": "string",

"name": "\_assetHash",

"type": "string"

},

{

"internalType": "address",

"name": "\_newOwner",

"type": "address"

}

],

"name": "transferOwnership",

"outputs": [],

"stateMutability": "nonpayable",

"type": "function"

},

{

"inputs": [

{

"internalType": "string",

"name": "\_assetHash",

"type": "string"

}

],

"name": "unpublishAsset",

"outputs": [],

"stateMutability": "nonpayable",

"type": "function"

}

]

if (!window.ethereum) {

alert('Meta Mask Not Found')

window.open("https://metamask.io/download/")

}

export const provider = new ethers.providers.Web3Provider(window.ethereum);

export const signer = provider.getSigner();

export const address = "0x7e988fD75a03EBde8C7699517D5e50274Bff30ed"

export const contract = new ethers.Contract(address, abi, signer)

**SOLUTIONING:**

The const { ethers } = require("ethers"); statement imports the ethers.js library, which is used for Ethereum-related operations.

The abi variable contains the Application Binary Interface (ABI) of the smart contract. The ABI is a JSON representation of the contract's functions, events, and data types.

The if (!window.ethereum) { ... } block checks if the MetaMask browser extension is installed. If it's not found, an alert is shown, and the user is redirected to the MetaMask download page.

The provider, signer, and address variables are used to connect to the Ethereum network and interact with the smart contract. The provider is created using the Web3Provider class from ethers.js, and the signer is obtained from the provider.

The contract variable is an instance of the smart contract, created using the ethers.Contract class. It represents the contract at a specific address and with a specific ABI.

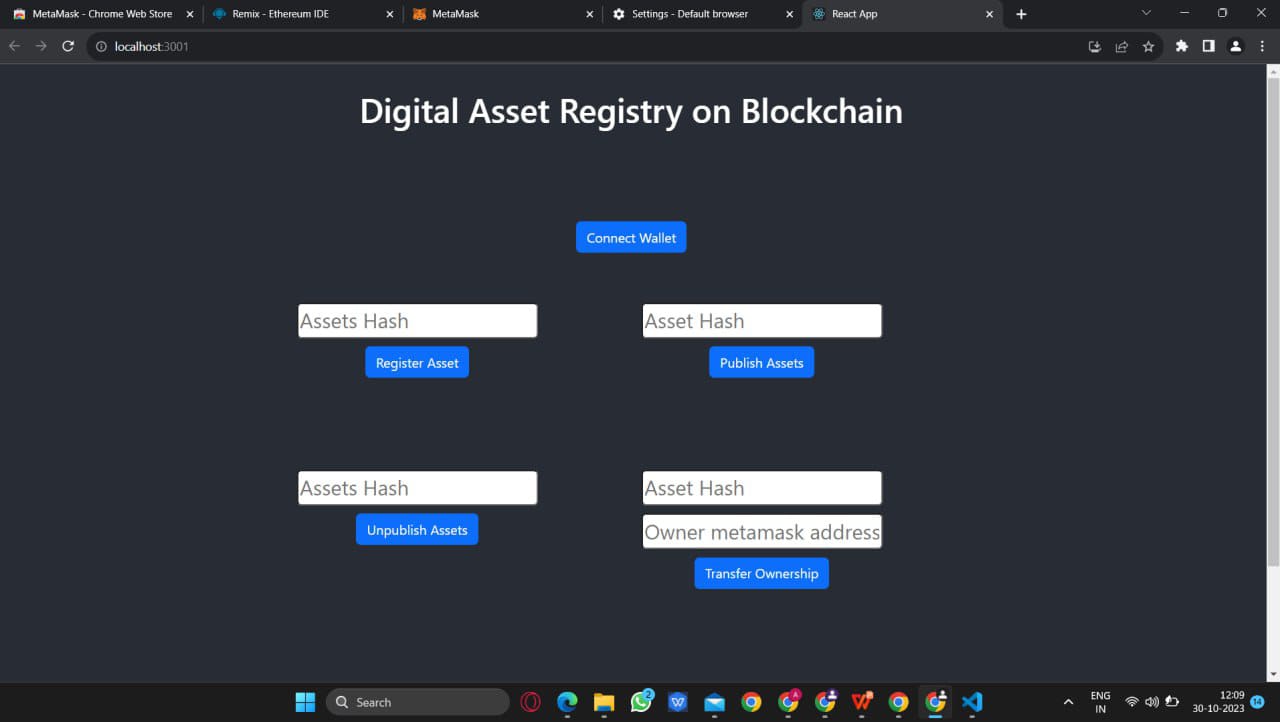
The code is designed to be used in a web application with MetaMask installed. It connects to the Ethereum network, sets up a provider, signer, and contract instance, allowing the application to interact with the smart contract.

**8.1 PERFORMANCE TESTING**

|  |  |  |
| --- | --- | --- |
| S.NO | PARAMETERS | SCREENSHOT |
| 1 | Information gathering |  |
| 2 | Extract the zip files |  |
| 3 | Remix IDE exploration |  |
| 4 | Open file explorer |  |
| 5 | Local host  (IP address) | **Local:**  **On Your Network:**  **http://localhost:3001**  [**http://192.168.156.250:3001**](http://192.168.156.250:3001) |

9. **RESULT:**

**9.1 OUTPUT SCREENSHOT**



**10. ADVANTAGES AND DISADVANTAGES:**

Digital asset management through Ethereum, which typically involves creating and managing tokens on the Ethereum blockchain, has its advantages and disadvantages.

**Advantages:**

* Security: Ethereum's blockchain is known for its robust security features, making it a trustworthy platform for digital asset management. Smart contracts provide tamper-proof rules for asset transfers and ownership.
* Transparency: Transactions on the Ethereum blockchain are publicly verifiable, which enhances transparency, and trust. This is particularly important for financial and asset management applications.
* Interoperability: Ethereum's ERC-20 and ERC-721 token standards have become industry standards, making it easier for assets to be traded and integrated into various applications and platforms.
* Decentralization: Ethereum operates on a decentralized network, reducing the risk of single points of failure. This makes it more resilient against censorship and fraud.
* Accessibility: Ethereum-based assets can be accessed and managed from anywhere with an internet connection, making it a convenient option for global asset management.

**Disadvantages:**

* Scalability: Ethereum has faced scalability issues, leading to congestion and high gas fees during periods of heavy network activity. This can be a drawback for managing assets efficiently.
* Complexity: Developing and managing smart contracts on Ethereum can be complex and requires expertise in blockchain development. Mistakes in contract code can lead to irreversible loss of assets.
* Regulatory Uncertainty: The regulatory environment for digital assets is still evolving, which can lead to uncertainty and potential legal challenges, especially if managing assets in a compliant manner is a priority.
* Energy Consumption: Ethereum, like many blockchain networks, consumes a significant amount of energy, which has raised environmental concerns.
* User Experience: While blockchain technology is advancing, the user experience for managing digital assets can still be challenging for non-technical users.

**11.CONCLUSION**:

In summary, the Digital Asset Management project, employing blockchain technology, represents a significant stride toward overcoming the challenges in handling digital assets securely and efficiently. By using a tailored blockchain system and the user-friendly Visual Studio Code environment, the project ensures a safe and decentralized approach to asset management. The incorporation of Remix IDE and IPFS simplifies the development and storage processes, while features like MetaMask enhance user security. The project not only meets current needs for a reliable asset management system but also sets the stage for scalability and adaptability, ensuring it stays relevant amid evolving technological landscapes. In essence, this initiative promises a secure and user-friendly solution for managing digital assets, paving the way for future advancements in the field.

**12. FUTURE SCOPE:**

Digital asset management on blockchain platforms like Ethereum has a promising future with key developments:

* Tokenization of Real-World Assets: Blockchain enables the representation and trading of real-world assets, enhancing accessibility and divisibility of traditionally illiquid assets like real estate and art.
* DeFi and Financial Services: Ethereum-based decentralized finance (DeFi) disrupts traditional financial services, expanding opportunities for digital asset management, lending, and trading.
* NFTs and Digital Collectibles: Non-fungible tokens (NFTs) extend beyond digital art, representing ownership of various assets and broadening digital asset management.
* Supply Chain and Inventory Management: Blockchain improves supply chain authenticity and traceability, impacting inventory and asset management.
* Interoperability and Cross-Chain Solutions: Projects facilitate asset movement between blockchains, increasing flexibility beyond Ethereum.
* Regulatory Developments: Clearer regulatory frameworks ease digital asset integration into the mainstream financial sector.
* Evolving Blockchain Technologies: Advancements address scalability and energy concerns, enhancing digital asset management efficiency.
* User-Friendly Solutions: Improved user interfaces and wallets enhance accessibility, making digital asset management more user-friendly.
* Integration with IoT: Combining blockchain and IoT streamlines physical asset management and automated tracking.
* Innovation in Smart Contracts: Smart contract advancements lead to more sophisticated and customizable asset management solutions.

**Source Code:**

Drive link of the source code file: <https://drive.google.com/file/d/1EcRTLuhwfzK1gSi7pFBnQ-YFrKhFZ8Gh/view>

GitHub Link

Demo Video Link