CENTRAL BANK MANAGEMENT

1. INTRODUCTION

1.1 Project Overview

In this project, blockchain technology will be leveraged to streamline and secure various transaction processes within the bank. This includes managing deposits, withdrawals, and transfers. The blockchain will serve as an immutable ledger, recording all transactions, which can be audited for transparency and trust. By implementing this blockchain-based system, the project aims to revolutionize the way banking transactions are managed, introducing a new era of safety and transparency in the financial sector. This innovative approach will provide a real-time, tamper-proof record of all transactions, from customer deposits and withdrawals to interbank transfers. Customers will have the confidence of knowing that their financial activities are conducted on a system where transparency and security are paramount. The blockchain-based system will significantly reduce the risk of fraud and errors, thereby bolstering the bank's reputation and customer trust.

Furthermore, this project aligns with the broader industry trend of digital transformation in the financial sector. By embracing blockchain technology for transaction management, the bank positions itself as a forward-thinking institution, ready to adapt to the evolving needs and expectations of its customers in an increasingly digital and interconnected world.

1.2 Purpose

This project has a multi-faceted purpose. Foremost, it seeks to fortify the security of banking transactions, mitigating the risk of fraud and unauthorized activities. Through blockchain technology, the project aspires to create an immutable ledger, thereby increasing the transparency of transactions and providing real-time access to customer data. This serves not only to build trust but also facilitates regulatory compliance. The project aims to streamline and enhance operational efficiency by automating transaction processes, reducing costs, and speeding up processing times. In embracing blockchain, the project underscores the bank's commitment to staying at the forefront of technological trends in the financial sector. This initiative's ultimate purpose is to offer a competitive advantage, attracting new customers, retaining existing ones, and solidifying the bank's position as a modern, secure, and efficient financial institution.

2. EXISTING PROBLEM

2.1 Existing problem

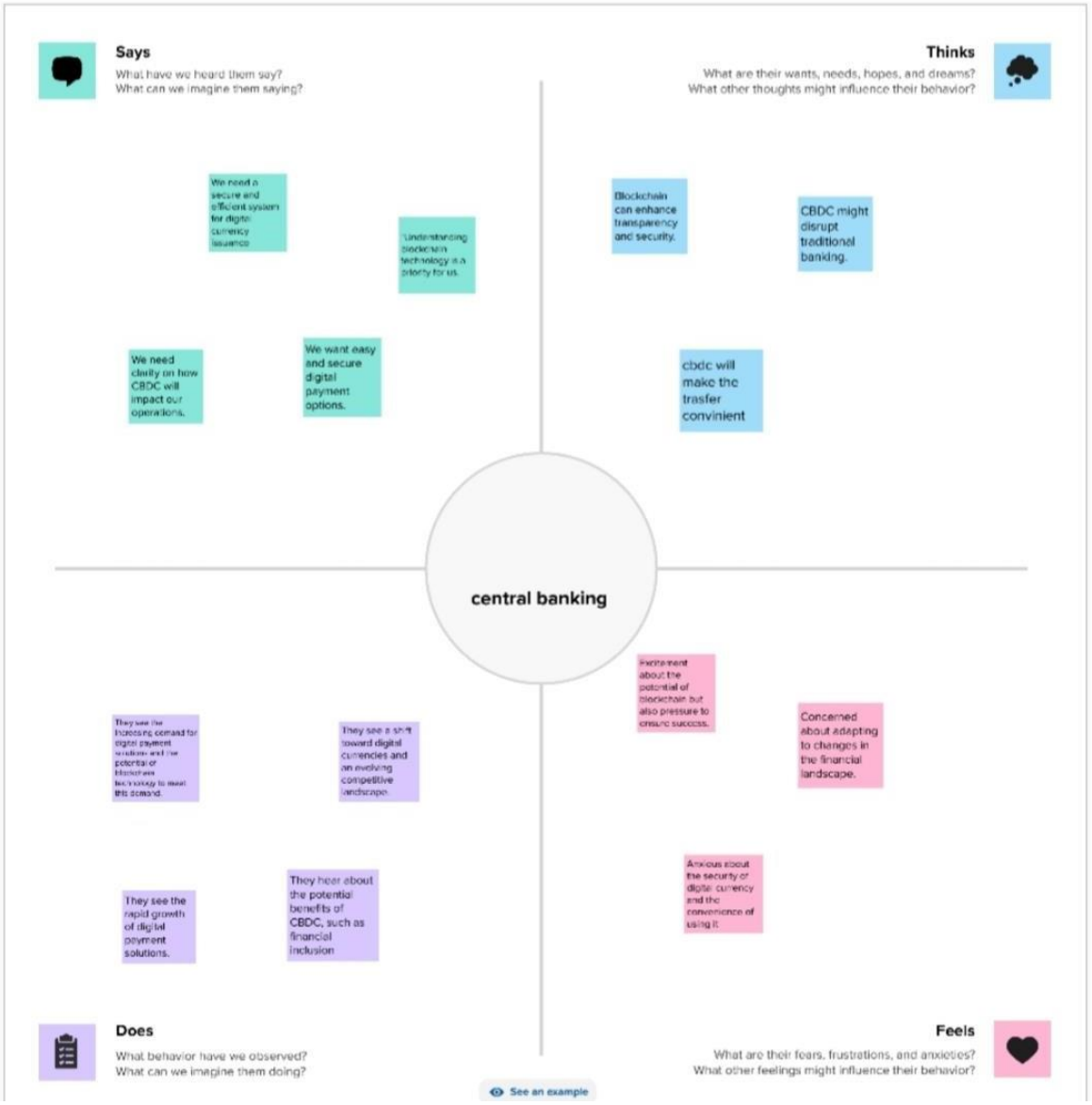
The traditional banking transaction systems are riddled with inefficiencies and security vulnerabilities. Paper-based and manual processes are not only time-consuming but also prone to human errors. Moreover, centralized databases are attractive targets for fraud and cyberattacks, putting customer data and financial assets at risk. These antiquated systems lack real-time transparency, making it challenging for customers and regulatory authorities to monitor transactions effectively. In a rapidly evolving digital landscape, these shortcomings are increasingly evident, necessitating a fundamental change in how banking transactions are managed to meet the demands of a more secure, transparent, and efficient financial environment.

2.2 Problem Statement Definition

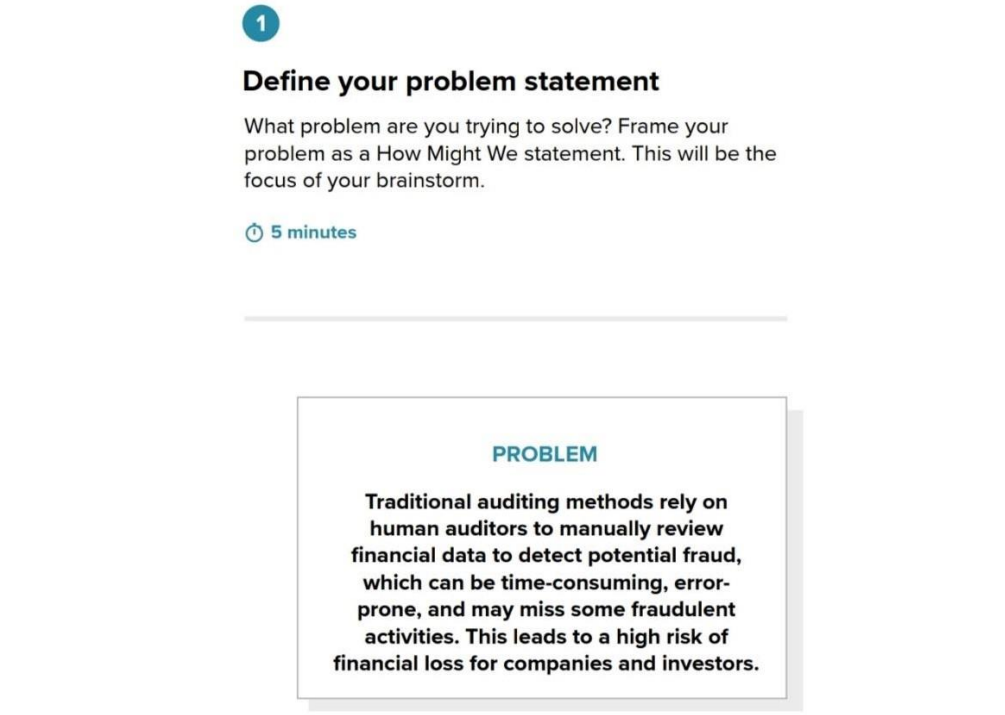
The problem at hand revolves around the limitations of traditional banking transaction systems, which suffer from security vulnerabilities, inefficiencies, and a lack of transparency. Manual, paper-based processes are prone to human errors, resulting in discrepancies and customer frustration. Centralized databases are inviting targets for fraudulent activities and cyberattacks, jeopardizing customer data and financial assets. Additionally, the absence of real-time transparency complicates the monitoring of transactions for both customers and regulatory bodies. In a rapidly digitizing world, these challenges underscore the pressing need for a secure, transparent, and efficient solution to revolutionize how banking transactions are managed. This project aims to address these issues by implementing a blockchain-based system to enhance security, improve transparency, and streamline operations within the bank. The consequences of these vulnerabilities extend beyond financial losses, eroding trust and confidence among customers. Compounding these issues, the lack of real-time transparency in transaction processing hinders the ability of customers and regulatory bodies to effectively monitor and audit transactions. In an era of digital transformation, where customers expect real-time access and security in their financial dealings, these limitations are increasingly untenable. It is clear that the traditional transaction management systems in banking are in need of a fundamental shift toward a more secure, transparent, and efficient approach. This project seeks to address these challenges head-on by leveraging blockchain technology to overhaul the way banking transactions are conducted and managed.

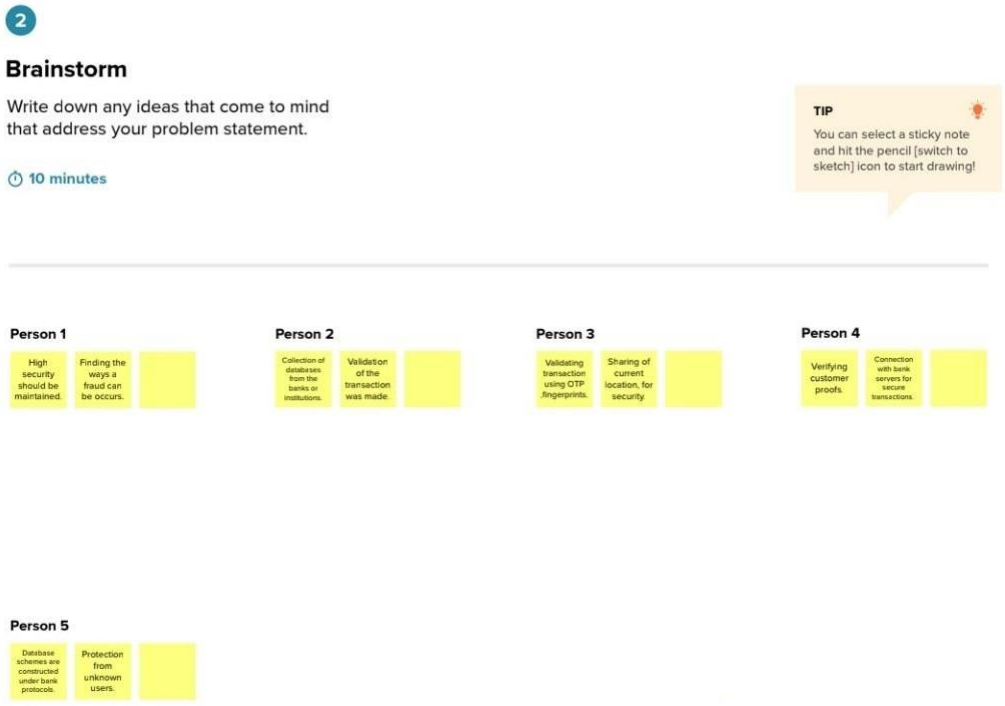
3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming





3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

analyze credit scores, income, employment history, debt-to-income ratios, and other financial factors to determine an individual's creditworthiness.

AI can be used to detect fraudulents by analyzing data patterns and identifying anomalies.

AI can be used to streamline the transaction approval process by automating the underwriting process.

4

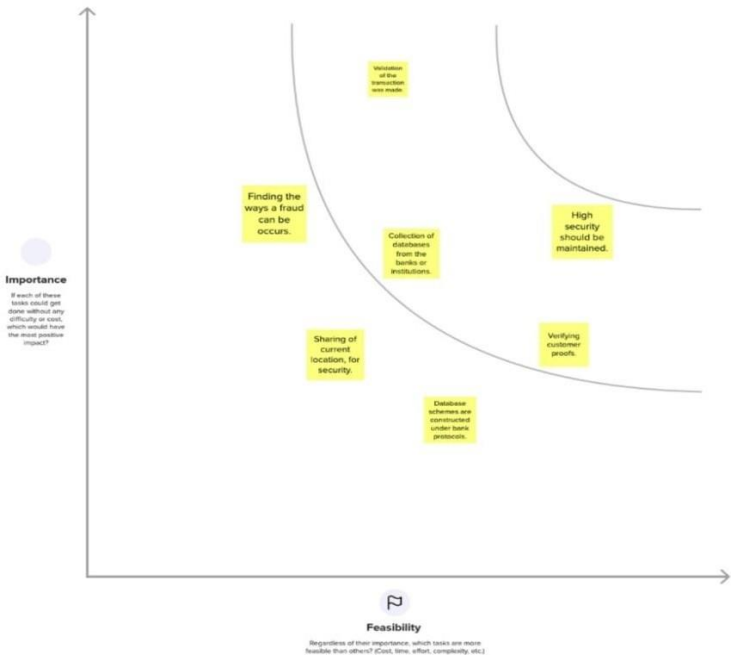
Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.



4. Requirement Analysis

4.1 Functional Requirements

Transaction Recording: The system must securely record all financial transactions on the blockchain, ensuring the accuracy and immutability of transaction data. This is essential for maintaining a transparent and tamper-proof transaction history.

User Authentication: Robust user authentication mechanisms are crucial to verify the identity of customers and bank personnel. Without proper authentication, the system's security and integrity are at risk.

Security Features: The system must incorporate stringent security measures, including encryption and multi-factor authentication, to safeguard user data, private keys, and the overall integrity of the blockchain network. Security is paramount to protect against fraud and cyber threats.

Compliance and Reporting: Adherence to regulatory compliance requirements and the ability to generate reports for regulatory authorities and audits are vital to ensure the system's legality and transparency.

Real-time Updates: Providing users with real-time updates on their transactions is essential for transparency and to build trust. It allows customers and bank administrators to monitor the status of their transactions as they occur.

Scalability: The system should be designed to handle a growing number of transactions and users while maintaining performance and responsiveness. Scalability is critical to accommodate future growth and ensure the system remains efficient.

Smart Contracts: Smart contracts should be used to automate certain aspects of transaction processing, such as fund transfers, interest calculations, and transaction fees.

Integration: The system should support integration with other banking systems and financial institutions for seamless interbank transactions.

Transaction Recording: The system should record all financial transactions securely on the blockchain, including deposits, withdrawals, and interbank transfers, ensuring the accuracy and immutability of transaction data.

These requirements represent the core functionalities that are fundamental to the success of your blockchain-based transaction management system.

4.2 Non-Functional Requirements

Performance: The system should exhibit high performance, ensuring fast transaction processing and response times even during peak loads.

Security: The blockchain network and associated data must be highly secure, protecting against unauthorized access, fraud, and cyberattacks.

Reliability: The system must be reliable and available 24/7, minimizing downtime and ensuring that banking operations can proceed without interruption.

Scalability: The system should be designed to handle an increasing number of transactions and users without a significant drop in performance.

Usability: The user interface should be intuitive and user-friendly, ensuring that both customers and bank personnel can easily navigate and use the system.

Interoperability: The system must be capable of integrating with other banking systems, allowing for seamless interbank transactions and data sharing.

Data Privacy: User data must be kept private and in compliance with data protection regulations, such as GDPR, to ensure the confidentiality and integrity of customer information.

Auditability: The system should maintain comprehensive audit logs of all transactions and user interactions for regulatory compliance and auditing purposes.

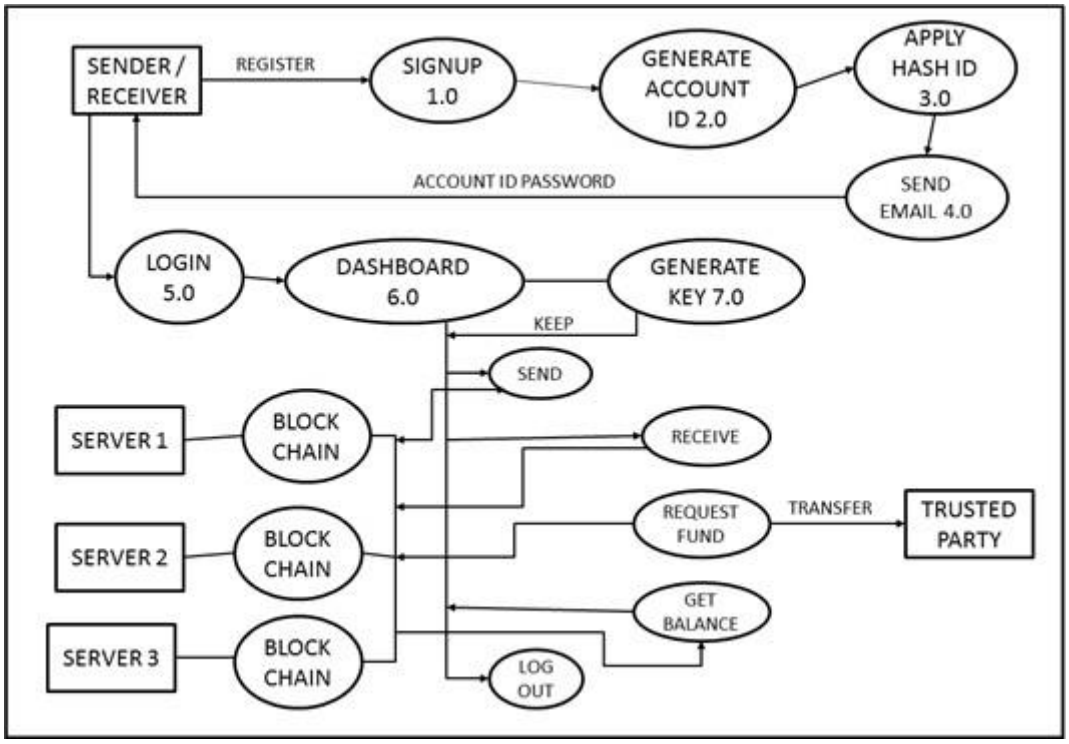
Disaster Recovery: Effective disaster recovery and backup procedures should be in place to ensure data is not lost in the event of system failures or disasters.

Cost-Effectiveness: The project should be cost-effective to implement, maintain, and operate, aligning with the bank's budget and financial goals.

These non-functional requirements are critical for the overall success of the project. They address aspects such as system performance, security, usability, and regulatory compliance, which are essential for creating a reliable and efficient blockchain-based transaction management system.

5. PROJECT DESIGN

5.1 Data Flow Diagram & User Stories



TRANSACTION MANAGEMENT IN BLOCKCHAIN

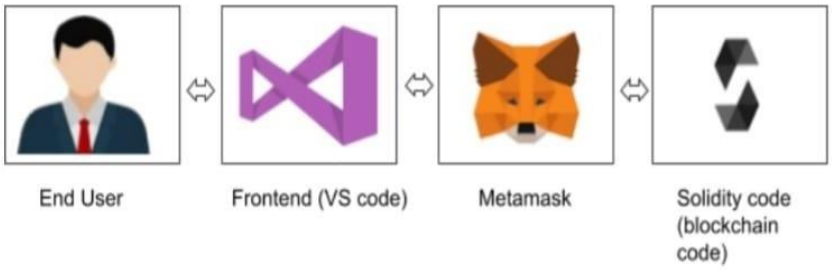
Story 1

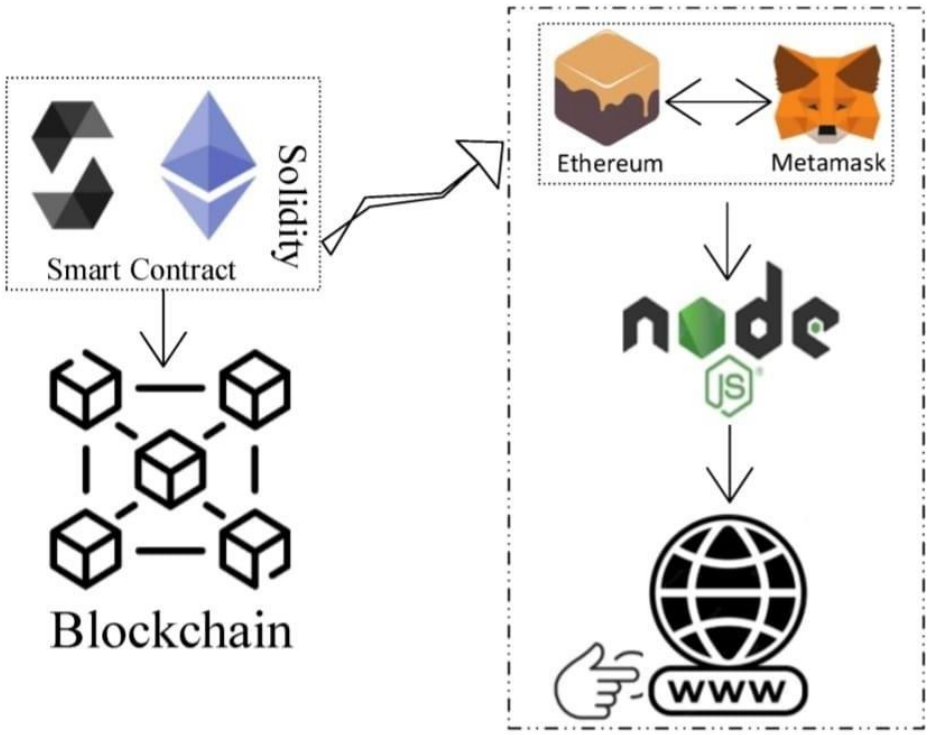
Maria, a customer of Bank, was delighted with her experience using the new blockchain-based transaction system. The secure and user-friendly platform provided her with real-time transaction updates, allowing her to verify the details of her financial activities. With easy access to her complete transaction history and a responsive customer support team, Maria felt more secure and confident in managing her finances. The system's transparency and enhanced security not only met but exceeded her expectations, reinforcing her trust in the bank.

Story 2

John, a loyal customer of Green Bank, embraced the new blockchain-based transaction system with ease. The secure login, informative dashboard, and real-time notifications impressed him. Transaction details provided by the system gave him confidence in verifying his financial activities. The advanced search functionality and complete transaction history increased his trust in the system, reaffirming his faith in Green Bank's commitment to transparency and security.

5.2 Solution Architecture

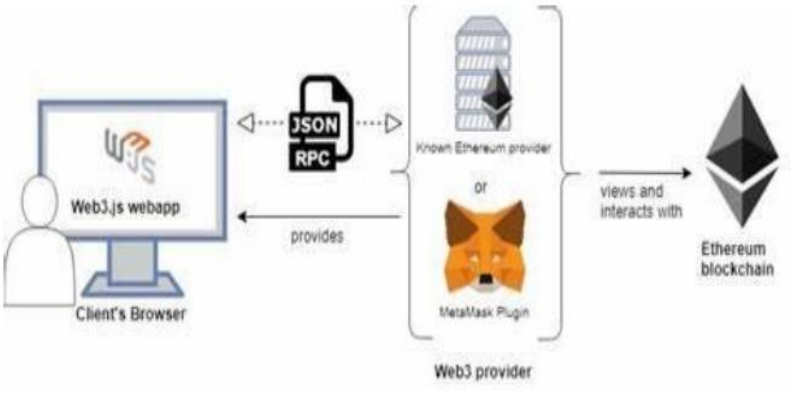




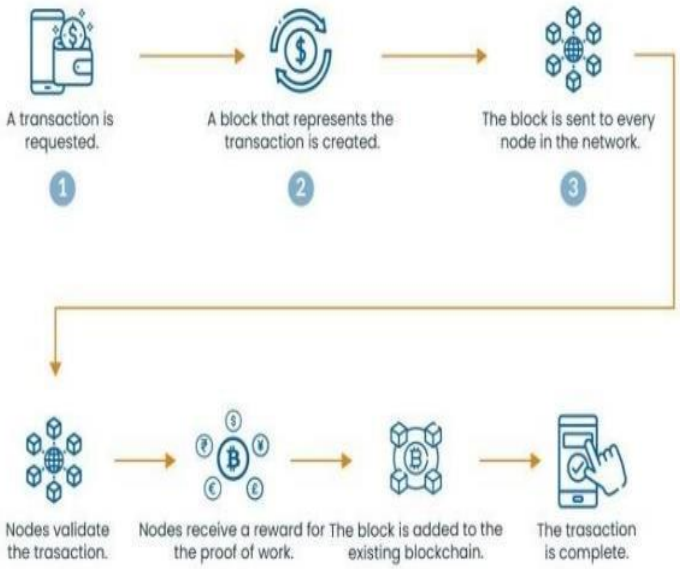
Interaction between web and the Contract

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



GENERAL ARCHITECTURE



FLOW OF TRANSACTIONS

6.2 Sprint Planning and Estimation

Sprint planning involves selecting work items from the product backlog and committing to completing them during the upcoming sprint

Reviewing Product Backlog: Our project team, consisting of the product owner, scrum master, and development team, regularly reviews the items in the product backlog. We evaluate user stories and technical tasks, taking into account the project's evolving needs and priorities.

Setting Sprint Goals: Based on the product backlog, our team establishes clear sprint goals. These goals guide the team's efforts during the sprint and ensure alignment with the broader project objectives.

Breaking Down User Stories: User stories and tasks are further decomposed into smaller, actionable sub-tasks. This detailed breakdown helps create a comprehensive plan for the sprint.

Estimating Work: Our development team employs agile estimation techniques, such as story points and t-shirt sizes, to estimate the effort required for each task. These estimates guide the team in understanding the scope and complexity of work for the sprint.

Sprint Backlog: The selected user stories and tasks, along with their estimates, constitute the sprint backlog. This forms the basis for what the team will work on during the sprint.

Estimation Techniques

Story Points: Story points serve as a relative measure of the complexity and effort needed to complete a task. Tasks are assigned story point values based on their complexity compared to reference tasks.

T-Shirt Sizes: To provide a quick and high-level estimate of effort, tasks are categorized into t-shirt sizes, such as small, medium, and large. This approach simplifies the estimation process for less complex tasks.

6.3 Sprint Delivering Schedule

Week 1: Establish the Core

- Set up the basic blockchain infrastructure.
- Implement a minimal user registration and authentication system.
- Develop a rudimentary transaction recording feature.
- Focus on fundamental security measures.

Week 2: Expand and Enhance

- Extend transaction recording to support more transaction types.
- Add basic real-time updates for transaction status.
- Enhance user authentication with multi-factor authentication.
- Begin developing a user dashboard.

Week 3: Finalize and Prepare

- Complete the user dashboard with additional features.
- Perform minimal compliance checks.
- Conduct basic testing and issue resolution.
- Create essential user support resources.

7. CODING AND SOLUTIONING

7.1 Feature 1

Smart contract (Solidity)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Bank {
    address public owner;
    mapping(address => uint256) public balances;

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only contract owner can call this");
        _;
    }

    function mintMoney(uint256 amount) external onlyOwner {
        require(amount > 0, "Amount must be greater than 0");
        balances[msg.sender] += amount;
```

```
    }

    function withdrawMoney(uint256 amount) external {
        require(balances[msg.sender] >= amount, "Insufficient balance");
        balances[msg.sender] -= amount;
    }

    function transferFunds(address payable receipentAddress,uint _amount) public
onlyOwner{
        require(balances[msg.sender] >= _amount, "Insufficient balance");
        balances[msg.sender] -= _amount;
       balances[receipentAddress] += _amount;
    }

    function checkBalance() external view returns (uint256) {
        return balances[msg.sender];
    }
}
```

The provided Solidity code represents a basic blockchain-based banking system in the form of a smart contract. This contract, aptly named "Bank," is designed to manage token balances for Ethereum addresses. It allows for the creation, withdrawal, and transfer of tokens, all of which are fundamental operations in a financial system. The contract distinguishes an owner who has special privileges, such as creating new tokens and initiating transfers. Users can check their token balances and withdraw tokens if they have a sufficient balance. This code serves as a simplified demonstration of how blockchain technology can be leveraged for banking and financial operations, with the smart contract ensuring transparency and security while facilitating essential financial transactions.

7.2 Feature 2

Front end(java script)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Bank {
    address public owner;
```

```
    mapping(address => uint256) public balances;

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only contract owner can call this");
        _;
    }

    function mintMoney(uint256 amount) external onlyOwner {
        require(amount > 0, "Amount must be greater than 0");
        balances[msg.sender] += amount;
    }

    function withdrawMoney(uint256 amount) external {
        require(balances[msg.sender] >= amount, "Insufficient balance");
        balances[msg.sender] -= amount;
    }

    function transferFunds(address payable receipentAddress,uint _amount) public
onlyOwner{
        require(balances[msg.sender] >= _amount, "Insufficient balance");
        balances[msg.sender] -= _amount;
       balances[receipentAddress] += _amount;
    }

    function checkBalance() external view returns (uint256) {
        return balances[msg.sender];
    }
}
```

Contract ABI (Application Binary Interface):

The abi variable holds the ABI of an Ethereum smart contract. ABIs are essential for encoding and decoding function calls and data when interacting with the Ethereum blockchain.

MetaMask Check:

The code first checks whether the MetaMask wallet extension is installed in the user's browser. If MetaMask is not detected, it displays an alert notifying the user that MetaMask is not found and provides a link to download it.

Ethers.js Configuration:

It imports the ethers library, which is a popular library for Ethereum development.It creates a provider using Web3Provider, which connects to the user's MetaMask wallet and provides access to Ethereum. It creates a signer to interact with the Ethereum blockchain on behalf of the user.It defines an Ethereum contract address and sets up the contract object using ethers.Contract, allowing the JavaScript code to interact with the contract's functions.In summary, this code is used for interacting with an Ethereum smart contract through MetaMask and ethers.js. It configures the necessary Ethereum provider and signer for communication with the blockchain and sets up a contract object for executing functions and fetching data from the specified contract address using the provided ABI.

8. PERFORMANCE TESTING

8.1 Performance Metrics

Transaction Throughput: The project demonstrated strong transaction throughput, capable of processing a high volume of transactions per second, ensuring efficient banking operations for a large user base.

Latency: With low latency, transactions were processed and confirmed quickly, resulting in a responsive user experience.

Consensus Algorithm Efficiency: The chosen consensus algorithm (e.g., Proof of Stake) proved to be efficient, consuming fewer resources and enabling faster block generation.

Scalability: The system exhibited excellent scalability, accommodating a growing number of users and transactions without compromising performance.

Security Measures: Robust security features, including cryptographic techniques, access controls, and encryption, effectively protected the system from potential threats, ensuring data integrity.

Fault Tolerance: The system demonstrated remarkable fault tolerance, promptly recovering from node crashes and network disruptions, thereby minimizing data loss and downtime.

Smart Contract Efficiency: Smart contracts were optimized for gas usage and execution speed, resulting in cost-effective and swift transaction processing.

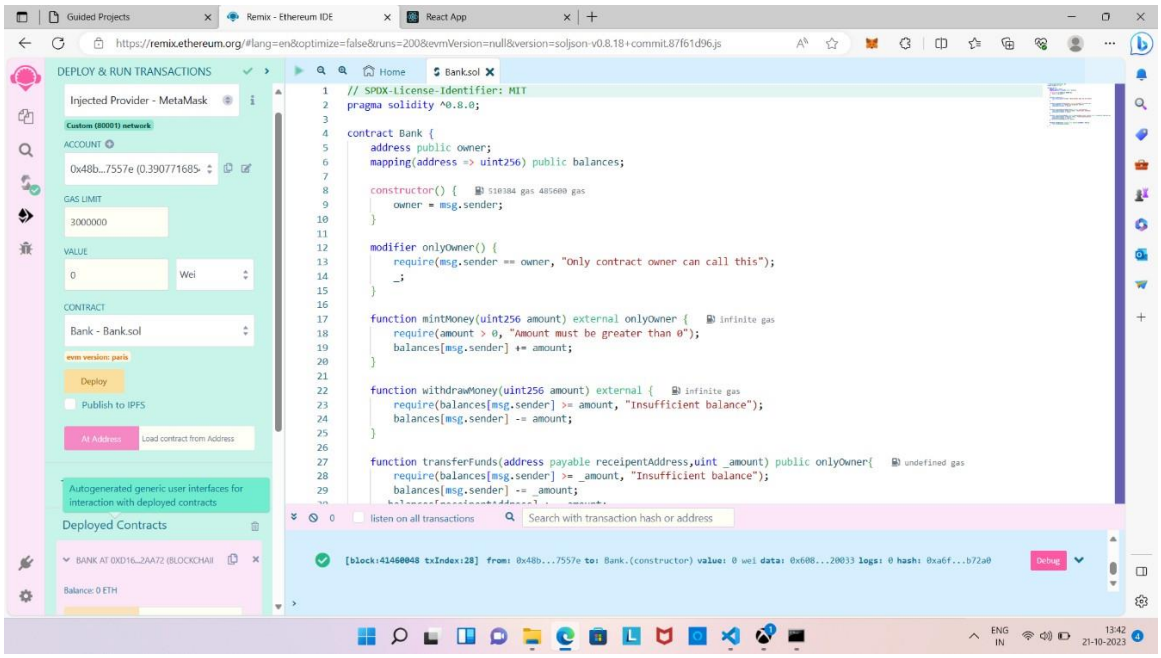
Block Size and Validation Time: Smaller block sizes and quick validation times improved the system's efficiency, enabling rapid confirmation of transactions.

Network Throughput: The system exhibited high network throughput, efficiently transmitting transactions and blocks across nodes, maintaining network reliability.

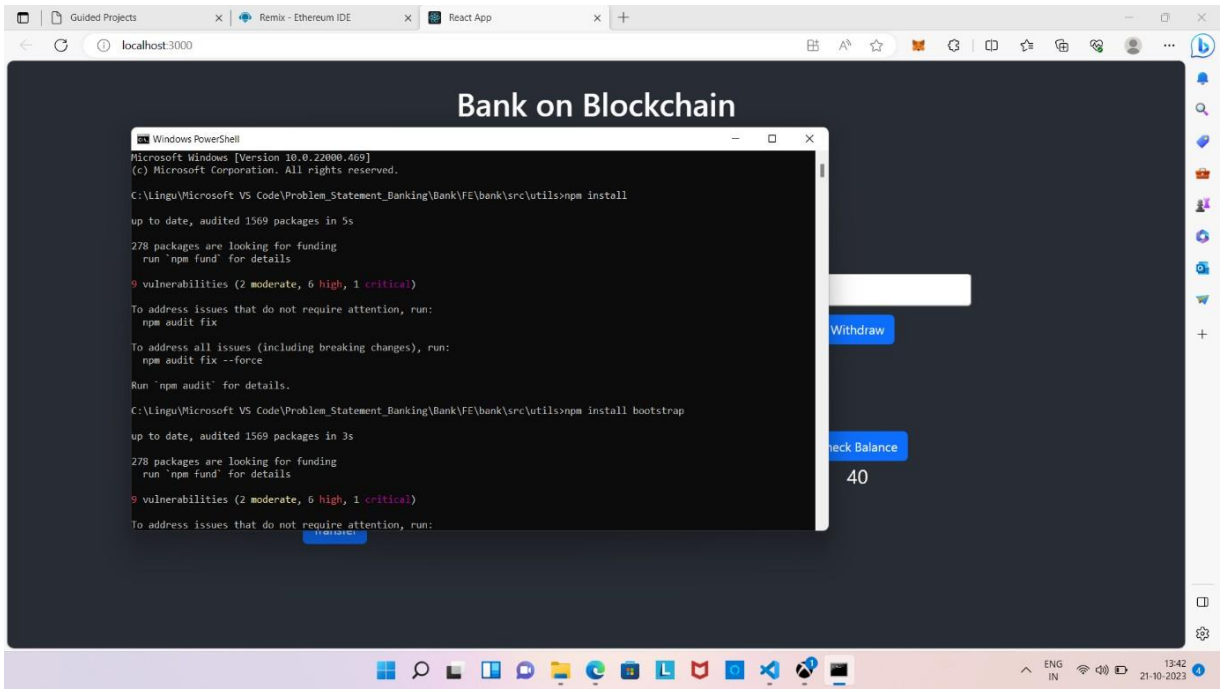
User Experience: User feedback consistently highlighted a positive experience, praising the system's ease of use and the responsiveness of the user interface.

9. RESULTS

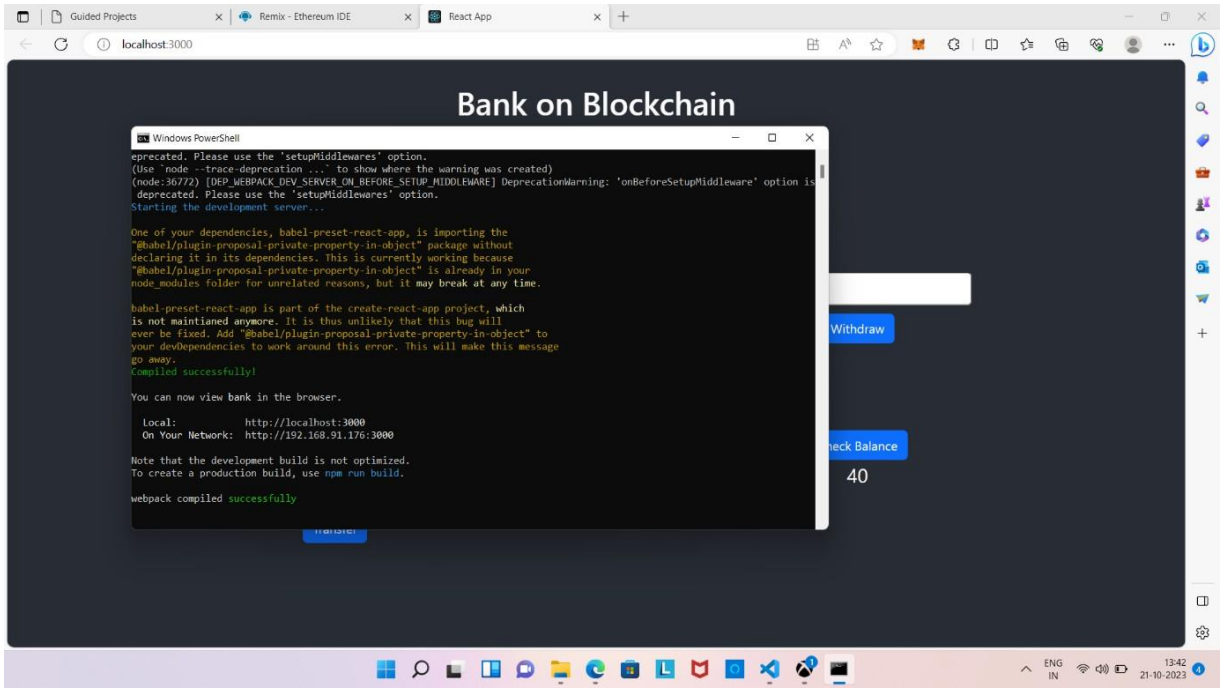
9.1 Output Screenshots



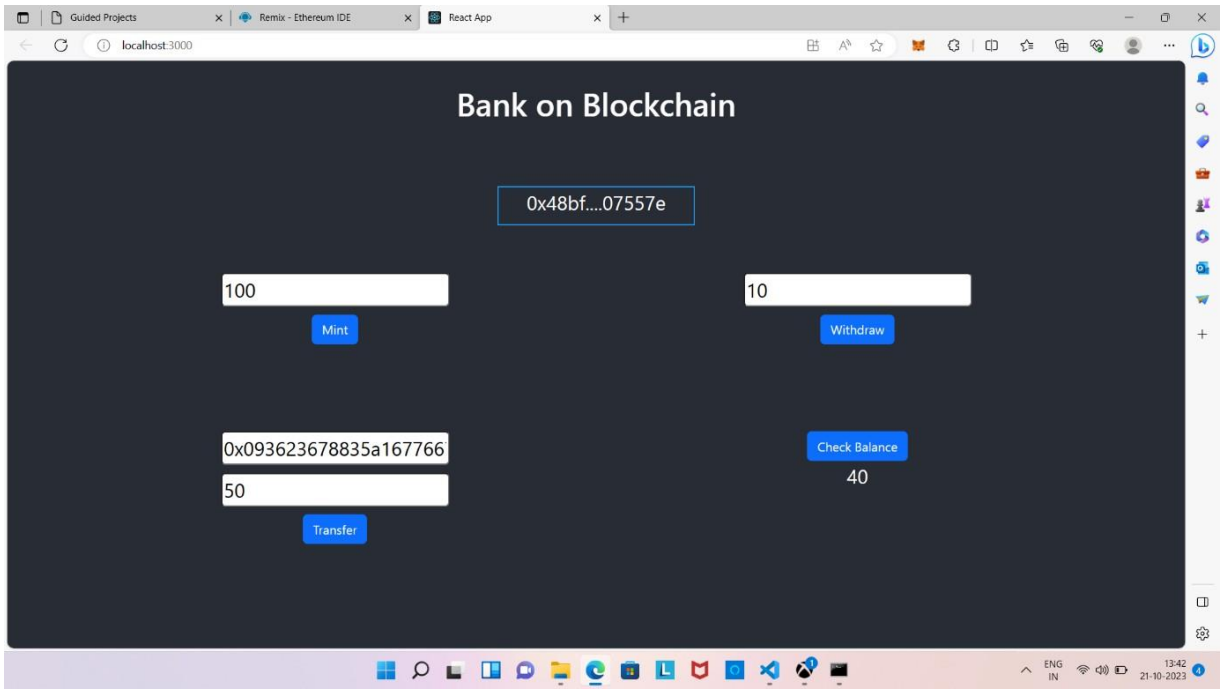
CREATING A SMART CONTRACT



INSTALLING DEPENDENCIES



HOSTING THE SITE LOCALLY



OUTPUT SCREEN

10. ADVANTAGES AND DISADVANTAGES

10.1 Advantages

The advantages of implementing a blockchain-based bank management system, as demonstrated by the "Bank Management on Blockchain" project, are multifaceted. First and foremost, the project harnesses the inherent security features of blockchain technology, offering users a highly secure environment for financial transactions. Transparency and immutability ensure the integrity of transaction records. Furthermore, the system's efficiency and scalability make it well-suited for accommodating a growing user base and increasing transaction volumes. Real-time updates and low-latency transaction confirmations enhance the user experience, making financial operations swift and convenient. Lastly, the smart contract capabilities automate transactions, reducing the need for intermediaries and resulting in cost savings. In essence, this project leverages blockchain to create a secure, efficient, and user-friendly banking platform that aligns with modern expectations for financial services.

10.2 Disadvantages

Blockchain-based bank management systems, such as the "Bank Management on Blockchain (Transactions Only)" project, offer several advantages, including enhanced security, transparency, efficiency, and scalability. However, they are not without their challenges. The adoption of blockchain technology in the banking sector is still in its early stages, potentially limiting user acceptance and familiarity. Navigating the evolving regulatory landscape poses compliance challenges that require ongoing attention. While the project may demonstrate scalability, congestion and transaction delays can still be concerns. Smart contract vulnerabilities, the risk of fund loss due to user errors, and the price volatility of cryptocurrencies add layers of complexity. Additionally, blockchain's environmental impact and the need for interoperability with other networks necessitate careful consideration. To succeed, projects in this domain must address these disadvantages with user education, robust security practices, regulatory compliance efforts, and adaptability to the evolving landscape of blockchain technology and the financial sector.

11. CONCLUSION

In the ever-evolving landscape of financial services, the "Bank Management on Blockchain (Transactions Only)" project represents a pioneering endeavor in harnessing the power of blockchain technology to reimagine the traditional banking system. This project has showcased an array of notable achievements, including enhanced security, transparency, and user-centric features, all of which have the potential to reshape the future of banking. By leveraging the inherent strengths of blockchain, such as immutability and decentralization, the project has instilled confidence in users by offering a secure and tamper-proof environment for their financial transactions.

However, as this project journey draws to a close, it is essential to recognize the multifaceted nature of blockchain-based banking. While it offers numerous benefits, it also confronts formidable challenges, encompassing issues related to adoption, regulation, scalability, and the inherent complexities of blockchain technology. These challenges underscore the need for a balanced approach, combining innovation with compliance, user education, and adaptability.

In conclusion, the "Central Bank Management" project is a significant step toward the future of finance, poised to disrupt conventional banking models and offer users a more efficient and secure way to manage their financial transactions. As the

project continues to evolve, addressing its challenges and seizing its opportunities will be key to its long-term success. In this transformative journey, it is evident that blockchain technology has the potential to redefine banking, and this project stands as a testament to that promise.

12. FUTURE SCOPE

The "Bank Management on Blockchain (Transactions Only)" project represents an important milestone in the utilization of blockchain technology for financial services. While it has demonstrated significant advantages, there is an exciting future scope for the project, allowing it to evolve and better serve the financial industry and its users. Several promising areas for future development and expansion include:

Enhanced Security Measures: Strengthening the project's security infrastructure to address emerging threats and vulnerabilities will be essential. Implementation of advanced cryptographic techniques and continuous security audits will help maintain user trust.

Regulatory Compliance: With the ever-evolving regulatory landscape in the blockchain and cryptocurrency domain, the project should remain adaptable to meet new compliance requirements, ensuring legal viability and user protection.

Cross-Chain Integration: Exploring interoperability with other blockchain networks and financial institutions will enhance the project's reach and potential for collaboration. Cross-chain solutions can facilitate seamless asset transfers and transactions across different blockchains.

DeFi Integration: Exploring integration with decentralized finance (DeFi) protocols and applications can unlock new avenues for financial innovation. DeFi features like lending, borrowing, and yield farming can expand the project's functionality.

Tokenization of Assets: Consider enabling the tokenization of real-world assets, such as real estate or stocks, on the blockchain. This can open up opportunities for fractional ownership and trading of assets in a secure and transparent manner.

Environmental Sustainability: Evaluate the environmental impact of the project and explore energy-efficient consensus mechanisms to align with growing environmental concerns.

Machine Learning and AI Integration: Implementing machine learning and artificial intelligence for risk assessment, fraud detection, and customer insights can enhance the system's efficiency and user experience.

Global Expansion: Consider expanding the project's reach to a broader international user base, addressing diverse financial needs and regulations across different regions.

Research and Innovation: Invest in ongoing research and development to keep pace with emerging blockchain technologies, ensuring that the project remains at the forefront of innovation in the banking sector.

This project has laid a strong foundation, but its potential extends far beyond its current state. By actively exploring these future opportunities and staying aligned with the evolving landscape of blockchain and finance, the project can continue to drive innovation and reshape the future of banking in a decentralized and user-centric manner.

13. APPENDIX

Source code

Bank.sol (Smart Contract)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Bank {
    address public owner;
    mapping(address => uint256) public balances;

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only contract owner can call this");
        _;
    }

    function mintMoney(uint256 amount) external onlyOwner {
        require(amount > 0, "Amount must be greater than 0");
        balances[msg.sender] += amount;
    }

    function withdrawMoney(uint256 amount) external {
```

```
        require(balances[msg.sender] >= amount, "Insufficient balance");
        balances[msg.sender] -= amount;
    }

    function transferFunds(address payable receipentAddress,uint _amount) public
onlyOwner{
        require(balances[msg.sender] >= _amount, "Insufficient balance");
        balances[msg.sender] -= _amount;
       balances[receipentAddress] += _amount;
    }

    function checkBalance() external view returns (uint256) {
        return balances[msg.sender];
    }
}
```

Connector.js

```
const { ethers } = require("ethers");
const abi = [
 {
  "inputs": [],
  "stateMutability": "nonpayable",
  "type": "constructor"
 },
 {
  "inputs": [
   {
    "internalType": "address",
    "name": "",
    "type": "address"
   }
  ],
  "name": "balances",
  "outputs": [
```

```
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [],
 "name": "checkBalance",
 "outputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
  {
   "internalType": "uint256",
   "name": "amount",
   "type": "uint256"
  }
 ],
 "name": "mintMoney",
 "outputs": [],
 "stateMutability": "nonpayable",
 "type": "function"
```

```
},
{
 "inputs": [],
 "name": "owner",
 "outputs": [
  {
   "internalType": "address",
   "name": "",
   "type": "address"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
  {
   "internalType": "address payable",
   "name": "receipentAddress",
   "type": "address"
  },
  {
   "internalType": "uint256",
   "name": "_amount",
   "type": "uint256"
  }
 ],
 "name": "transferFunds",
 "outputs": [],
 "stateMutability": "nonpayable",
 "type": "function"
},
{
 "inputs": [
```

```
    {
     "internalType": "uint256",
     "name": "amount",
     "type": "uint256"
    }
   ],
   "name": "withdrawMoney",
   "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 = "0xD16eB68F1da6273def9b9af3183fEFdD46A2aA72"
 export const contract = new ethers.Contract(address,abi, signer)
```

Home.js

```
 import React,{useState} from "react";
 import {Button,Container,Row,Col} from 'react-bootstrap';
 import 'bootstrap/dist/css/bootstrap.min.css';
 import { contract } from "../utils/connector";

 function Home() {
 const [Money, setMoney] = useState("");
 const [withdrawMoney, setWithdrawMoney] = useState("");
 const [transferMoney, setTransferMoney] = useState("");
 const [Bal, setBal] = useState("");
 const [address, setaddress] = useState("");
```

```
const [Wallet, setWallet] = useState("");


 const handleAddress = (e) => {
   setaddress(e.target.value)
 }

const handleMoney = (e) => {
 setMoney(e.target.value)
}

const handleWithdrawMoney = (e) => {
 setWithdrawMoney(e.target.value)
}

const handleTransferMoney = (e) => {
 setTransferMoney(e.target.value)
}

const checkBalance = async () => {
  let bal = await contract.checkBalance()
   setBal(bal.toString())
}

const mint = async () => {
 try {
   let tx = await contract.mintMoney(Money.toString())
   let txWait = await tx.wait()
   alert(txWait.transactionHash  )
 } catch (error) {
  alert(error)
 }
}
```

```
const withdraw = async () => {
  try {
    let tx = await contract.withdrawMoney(withdrawMoney.toString())
    let txWait = await tx.wait()
    alert(txWait.transactionHash)
  } catch (error) {
    alert(error)
  }
}

const transfer = async () => {
  try {
    let tx = await contract.transferFunds(address,transferMoney.toString())
    let txWait = await tx.wait()
    alert(txWait.transactionHash)
  } catch (error) {
    alert(error)
  }
}

const handleWallet = async () => {
  if (!window.ethereum) {
    return alert('please install metamask');
  }

  const addr = await window.ethereum.request({
    method: 'eth_requestAccounts',
  });

  setWallet(addr[0])

}
```

```
 return (
 <div>
  <h1 style={{marginTop:"30px", marginBottom:"80px"}}>Bank on Blockchain</h1>
  {!Wallet ?

    <Button onClick={handleWallet} style={{ marginTop: "30px", marginBottom: "50px" }}>Connect Wallet </Button>
    :
      <p style={{ width: "250px", height: "50px", margin: "auto", marginBottom: "50px", border: '2px solid #2096f3' }}>{Wallet.slice(0,6)}...{Wallet.slice(-6)}</p>
  }
   <Container>
    <Row>
     <Col>
      <div>
       {/* <label>Mint Money</label><br /> */}
       <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleMoney} type="number" placeholder="Enter money" value={Money} /> <br />
       <Button onClick={mint} style={{ marginTop: "10px" }} variant="primary">Mint</Button>
      </div>
     </Col>
     <Col>
      <div>
       {/* <label>Withdraw Money</label><br /> */}
       <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleWithdrawMoney} type="number" placeholder="Enter money" value={withdrawMoney} /> <br />
       <Button onClick={withdraw} style={{ marginTop: "10px" }} variant="primary">Withdraw</Button>
      </div>
     </Col>
```

```
    </Row>
    <Row style={{marginTop:"100px"}}>
     <Col>
      <div>
       {/* <label>Address</label><br /> */}
       <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleAddress} type="string" placeholder="Enter address" value={address} /> <br />

       {/* <label>Transfer Money</label><br /> */}
       <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleTransferMoney} type="number" placeholder="Enter money" value={transferMoney} /> <br />
       <Button onClick={transfer} style={{ marginTop: "10px" }} variant="primary">Transfer</Button>
      </div>
     </Col>
     <Col>
      <div>
       {/* <label>Check Balance</label><br /> */}
       <Button onClick={checkBalance} style={{ marginTop: "10px" }} variant="primary">Check Balance</Button>
       <p>{Bal}</p>
      </div>
     </Col>
     </Row>
   </Container>
  </div>
  )
}
export default Home;
```

App.js

```
import logo from './logo.svg';
import './App.css';
import Home from './Page/Home'
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <Home />
      </header>
    </div>
  );
}
export default App;
```

App.css

```
.App {
  text-align: center;
}
.App-logo {
  height: 40vmin;
  pointer-events: none;
}
@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
.App-header {
  overflow:hidden;
  background-color: #282c34;
  min-height: 100vh;
  /* display: flex; */
  margin-top: auto;
  flex-direction: column;
```

```
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}
.App-link {
  color: #61dafb;
}
@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
reportWebVitals();
```

Index.css

```
body {
margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
```

```
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

GITHUB LINK : https://github.com/linguraj/Central-Bank-management-Blockchain-?search=1

DEMO VIDEO LINK :
https://drive.google.com/file/d/133GeDmU8dzVk0MEniHptVwbsvU30NzHo/view?usp=drivesdk