MAILAM ENGINEERING COLLEGE

MAILAM-604304

CENTRAL BANK SMART CONTRACT IN BLOCK CHAIN TECHNOLOGY

PROJECT REPORT

Submitted by

TEAM ID: NM2023TM7D00537

TEAM MEMBERS :

VIGNESHWARAN .K - 421620114323

RITHICK RAGUL.V - 421620114318

LOGEESHWARAN. S - 421620114311

ASHWANTH.S - 421620114302

TABLE OF CONTENT

1.INTRODUCTION

1.1 PROJECT OVERVIEW

A Central Bank Smart Contract in a blockchain project typically involves the use of blockchain technology and smart contracts to enhance the operations and functions of a central bank. Here's an overview of such a project:Enhance transparency, security, and efficiency in central bank operations.Improve the traceability and accountability of financial transactions.Explore opportunities for automation in various financial processes.Settlement and clearing of interbank transactions.Issuance and management of digital currencies.Regulatory reporting and compliance.Auditing and monitoring financial processes.Blockchain Technology Selection:Choose a blockchain platform (e.g., Ethereum, Hyperledger, or a private blockchain) that suits the specific requirements and scalability needs of the central bank.Smart Contract Development:Develop smart contracts that automate and enforce predefined rules for various financial processes.Ensure robust security measures and code auditing to prevent vulnerabilities.Implement privacy-enhancing technologies to protect sensitive financial data.Set up strong authentication and access control mechanismWork closely with regulatory authorities to ensure that the smart contracts comply with financial regulations.Rigorous testing to identify and fix bugs, vulnerabilities, and ensure the reliability of smart contracts.Integrate the blockchain and smart contract solutions with the existing systems and processes of the central bank.Train the central bank's staff and relevant stakeholders on how to use and interact with the blockchain and smart contract systems.Implement a monitoring system to track the performance of the blockchain network and smart contracts.Regularly update and maintain the system to address evolving needs and security concerns.Collaborate with experts in blockchain technology and financial regulation.Engage in ongoing research to adapt to new developments in the field.Communicate with the public about the benefits and security of the digital currency or financial processes built on blockchain.Develop strategies for handling potential risks, including cyber threats and system failures.Regularly evaluate the project's success against predefined goals and gather feedback for continuous improvement.Implementing a central bank

smart contract on a blockchain is a complex and multifaceted project that requires careful planning, collaboration, and a strong focus on security and compliance. It has the potential to transform the central banking system and improve the efficiency and transparency of financial processes.

PURPOSE

The purpose of implementing a Central Bank smart contract in a blockchain project is to achieve several key objectives and benefits:Enhanced Transparency: Blockchain technology allows for transparent, immutable, and auditable records of financial transactions. The central bank can use smart contracts to make its operations more transparent to the public and stakeholders.Improved Security: Smart contracts on a blockchain are highly secure and resistant to tampering. This enhances the security of central bank operations, reducing the risk of fraud, cyberattacks, and unauthorized access to financial data.Efficiency and Automation: Smart contracts can automate various financial processes, reducing the need for manual intervention. This leads to increased operational efficiency, reduced processing times, and minimized errors.Cost Savings: Automation and reduced manual processes can lead to cost savings for the central bank. Fewer intermediaries are needed, and operational overhead can be reduced.Traceability and Accountability: All transactions on a blockchain are traceable and immutable. This means that the central bank can easily track the flow of funds and assets, improving accountability and fraud prevention.Digital Currency Issuance: Central banks can explore the issuance and management of digital currencies (Central Bank Digital Currencies or CBDCs) using blockchain technology. This can provide a more secure and efficient means of transacting in digital currency.Interbank Settlements: Blockchain-based smart contracts can be used for interbank settlements, providing a faster and more reliable way for financial institutions to settle transactions.Regulatory Compliance: The use of smart contracts can help central banks better enforce financial regulations, ensuring that financial institutions comply with the law and reporting requirements.Innovation and Research: Central banks can stay at the forefront of financial technology by exploring blockchain and smart contract solutions. This fosters innovation and research in the financial sector.Resilience: Blockchain networks are decentralized and resilient, which can be advantageous in ensuring the continuity of central bank operations, even in the face of disruptions.Public Confidence: Implementing secure and transparent

blockchain-based systems can build public confidence in the central bank's operations and digital currency initiatives.Reduced Counterparty Risk: Smart contracts can automate trustless transactions, reducing counterparty risk, and making financial operations more secure.In summary, the purpose of a Central Bank smart contract in a blockchain project is to leverage blockchain's capabilities to improve the central bank's operations, security, and transparency, as well as explore the potential of digital currencies while ensuring regulatory compliance and efficiency in the financial sector

LITERATURE SURWAY

EXISTING PROBLEM

Implementing a central bank's functions through a blockchain-based smart contract **sys**tem can face several challenges:Scalability: Managing an entire economy's financial transactions on a blockchain can be highly demanding in terms of scalability. Many blockchains have limited transaction throughput.Security: Ensuring the security of the smart contracts and the blockchain itself is critical. Vulnerabilities in the code can lead to significant financial losses.

Privacy: Maintaining the privacy of sensitive financial data while still allowing for transparency and auditability is a delicate balance.Legal and Regulatory Compliance: Central banks are subject to strict regulatory oversight. Adhering to these regulations in a blockchain-based system can be complex.Interoperability: Coordinating with other financial systems and institutions that may not be on the same blockchain can be challenging.Monetary Policy Implementation: Implementing monetary policies, such as controlling the money supply, can be difficult in a decentralized system.Consensus Mechanisms: Selecting an appropriate consensus mechanism (e.g., Proof of Work, Proof of Stake) can impact the system's efficiency and security.

User Adoption: Getting individuals and businesses to adopt the new system can be challenging. It requires trust and usability.Smart Contract Complexity: Writing complex financial smart contracts is inherently challenging and can lead to errors.Data Oracles: External data is often needed for financial decisions. Using oracles to fetch this data can introduce vulnerabilities.Operational Risk: Smart contracts are immutable once deployed. Fixing issues or bugs can be difficult and

costly.Resource Requirements: Running a central bank on a blockchain can require significant computational resources, which can be expensive.Addressing these challenges requires a deep understanding of both blockchain technology and central banking principles, as well as collaboration with experts in these fields

REFERENCE

Creating a central bank smart contract on a blockchain involves several key references and considerations. Here are some important points to reference when working on such a project:

1. Blockchain Platform: Choose a suitable blockchain platform, such as Ethereum, Binance Smart Chain, or a private blockchain, depending on your requirements.

2. Smart Contract Development: Reference the specific programming language used for smart contract development on your chosen platform (e.g., Solidity for Ethereum). You'll need to understand the language and its syntax.

3. Central Bank Functions: Define the functions and responsibilities of the central bank within the smart contract, including monetary policy, issuing currency, and regulating the financial system.

4. Token Standards: If you're creating a digital currency, reference the token standards for your blockchain platform (e.g., ERC-20 for Ethereum) and customize them to fit your central bank's requirements.

5. Monetary Policy Rules: Implement the rules and policies governing the money supply, interest rates, and other monetary tools used by the central bank.

6. Smart Contract Security: Reference best practices for smart contract security, including vulnerability detection and mitigation to ensure the central bank's functions are secure.

7.Oracles: Consider using oracles to connect your smart contract with real-world data, such as economic indicators, to make monetary decisions.

8. Legal and Regulatory Compliance: Ensure that your smart contract complies with local and international financial regulations and reference legal experts for guidance.

9. Governance and Access Control: Determine who has control over the central

bank's smart contract, and establish a governance model, which may include multi-signature wallets or decentralized autonomous organizations (DAOs).

10. Testing and Auditing: Thoroughly test the smart contract and consider third-party audits to identify and fix any vulnerabilities or bugs.

11. Scalability and Efficiency: Consider blockchain scalability solutions and optimize the smart contract for efficiency, as central bank functions can be resource-intensive.

12. Privacy and Confidentiality: Reference techniques for preserving the privacy and confidentiality of transactions, especially if dealing with sensitive financial data.

13. Stakeholder Engagement: Collaborate with financial institutions, regulators, and other stakeholders to ensure a smooth integration of the central bank's smart contract into the financial system.

PROBLEM STATEMENT DEFINITION

Introduction:

The central bank plays a pivotal role in a country's economy by controlling the money supply and interest rates. The use of blockchain technology can enhance the transparency, security, and efficiency of central bank operations. This project aims to define the problem statement for the development of a central bank smart contract on a blockchain.

2. Problem Description:

The central bank smart contract project seeks to address the following key challenges:

a. Monetary Policy Implementation: Design a smart contract that facilitates the implementation of monetary policies, such as setting interest rates, managing the money supply, and executing open market operations.

b. Interoperability: Ensure that the smart contract can seamlessly interact with the existing financial infrastructure, including commercial banks, payment systems, and government entities.

c. Security and Data Privacy: Implement robust security measures to protect sensitive financial data and ensure that transactions are private and tamper-proof.

d. Scalability: Develop a blockchain solution that can handle a high volume of transactions efficiently, especially during economic crises or rapid policy changes.

e. Regulatory Compliance: Ensure that the smart contract adheres to existing financial regulations and international standards while allowing for flexibility to adapt to changing regulatory environments.

f. Resilience and Disaster Recovery: Create mechanisms for data recovery and business continuity in the face of unexpected disruptions or cyberattacks.

3. Project Objectives:

The primary objectives of the project are as follows:

a. Design a smart contract that can execute and enforce central bank policies in a decentralized and automated manner.

b. Integrate the smart contract with the existing financial infrastructure, allowing for seamless data exchange and transactions.

c. Implement strong encryption and authentication mechanisms to protect sensitive financial data and ensure the privacy of transactions.

d. Develop a scalable blockchain solution that can handle a high volume of transactions efficiently.

e. Ensure that the smart contract remains compliant with financial regulations and standards, while allowing for updates and modifications as necessary.

f. Establish robust disaster recovery and business continuity measures to maintain operations in adverse conditions.

4. Expected Outcomes:

The successful development of the central bank smart contract project is expected to result in:

a. Improved efficiency in implementing and enforcing monetary policies.

b. Enhanced transparency and security in central bank operations.

c. Increased trust in the financial system through tamper-proof transactions.

d. Greater resilience in the face of financial crises and cyber threats.

e. Improved regulatory compliance and adaptability to changing regulations.

5. Stakeholders:

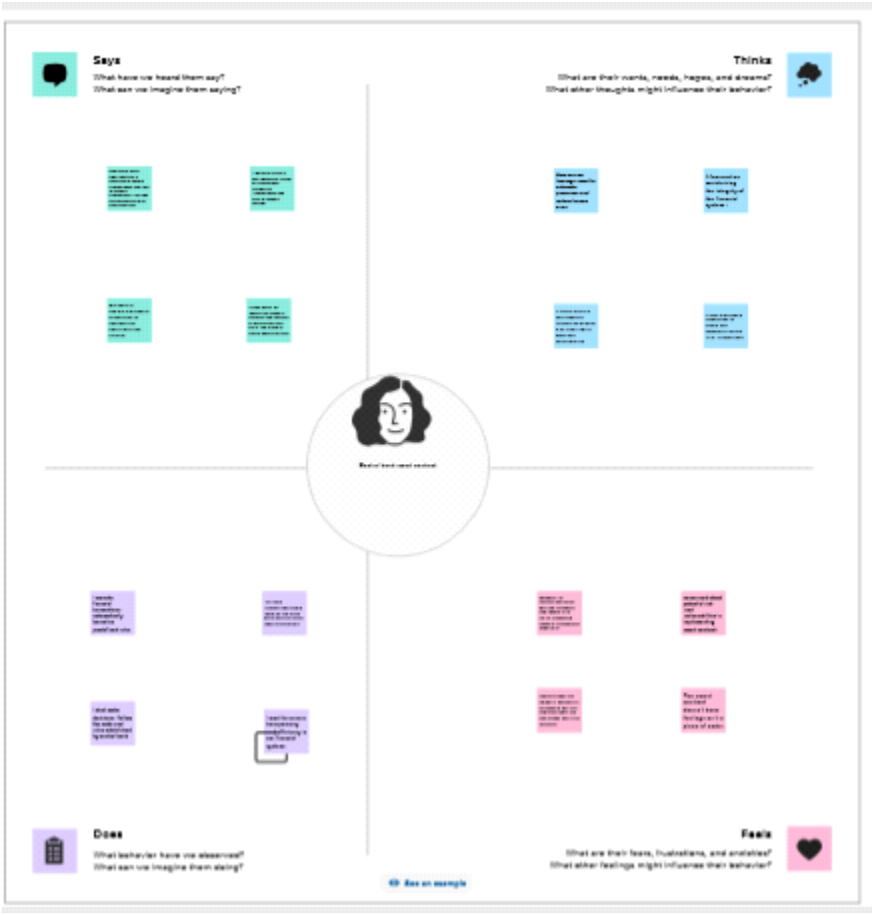
Key stakeholders in this project include central bank officials, blockchain developers, financial institutions, government agencies, and regulatory bodies.

6. Project Scope:

The project will focus on the development of the central bank smart contract and its integration with the existing financial ecosystem. It will not encompass broader monetary policy decisions**.**

3 IDEATION &PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 BRAIN STORMING

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

Functional requirements for a Central Bank smart contract in a blockchain project would typically include:

1. Token Creation and Management: The ability to create and manage digital tokens representing the national currency.

2. Transaction Processing: The contract should handle transactions securely and efficiently, including transfers, payments, and settlements.

3. Identity Verification: Incorporate identity verification measures to ensure that only authorized entities can transact with the smart contract.

4. Monetary Policy Enforcement: Implement rules and controls to ensure compliance with the central bank's monetary policy, such as interest rates and money supply.

5. Reserve Management: Enable the central bank to manage its reserves, including the issuance and redemption of tokens.

6. Regulatory Compliance: Ensure that the smart contract adheres to all relevant financial regulations and can adapt to changing regulatory requirements.

7. Data Transparency: Provide real-time data and auditability features to ensure transparency and accountability.

8. Interoperability: Facilitate interactions with other smart contracts or blockchain networks, including international payment systems.

9. Security Measures: Implement robust security features to protect against fraud, hacking, and unauthorized access.

10. Reporting and Analytics: Include reporting and analytics tools to monitor the performance and usage of the smart contract.

11. Governance Mechanisms: Define a governance structure to make necessary updates and changes to the contract while maintaining decentralization and security.

12. User Interface: Develop user-friendly interfaces for central bank administrators and possibly the public to interact with the contract.

13. Scalability: Ensure that the contract can scale to accommodate a large volume of transactions.

14. Smart Contract Upgradability: Include mechanisms for upgrading the smart contract while maintaining the integrity of the system.

15. Testing and Simulation: Extensive testing, including stress testing and simulation, to ensure the contract functions reliably in various scenarios.

16. Disaster Recovery: Develop a plan for disaster recovery and data backup to ensure continuity of operations in case of system failures.

17. Documentation: Comprehensive documentation to assist developers, auditors, and users in understanding and using the smart contract.

18. Compliance with Blockchain Standards: Adherence to relevant blockchain standards and best practices.

4.2 NON FUNCTIONAL REQUIREMENT

Non-functional requirements for a Central Bank smart contract in a blockchain project are essential to ensure the system's performance, security, and reliability. Here are some key non-functional requirements:

1. Security:

- The smart contract should be resistant to attacks, including code vulnerabilities and hacking attempts.

- Data confidentiality, integrity, and availability must be maintained.

- Compliance with relevant regulations, like KYC and AML, must be ensured.

2. Scalability:

- The contract should handle a large number of transactions efficiently.

- Scalability solutions, such as sharding or layer 2 solutions, may be necessary.

3. Performance:

- The contract should execute transactions quickly to meet the demands of a Central Bank.

- Low latency is critical for real-time financial operations.

4. Reliability:

- The smart contract should be highly available and minimize downtime.

- Backup and disaster recovery mechanisms should be in place.

5. Interoperability:

- Ensure that the contract can interact with other blockchain networks or legacy financial systems as required by the Central Bank.

6. Compliance:

- Adherence to legal and regulatory requirements is paramount, especially for a Central Bank project.

7. Auditability and Traceability:

- The contract should maintain an immutable history of all transactions for auditing and forensic purposes.

8. Usability:

- A user-friendly interface for Central Bank personnel to interact with the smart contract is important.

9. Data Management:

- Efficient storage and retrieval of data, especially for large-scale operations.

10. Cost-Efficiency:

- Optimize gas/transaction fees to reduce operational costs, especially in a public blockchain setting.

11. Resilience:

- The smart contract should be able to recover gracefully from failures, ensuring the Central Bank's operations are not severely impacted.

12. Adaptability:

- The ability to upgrade or modify the smart contract to meet changing requirements or address security concerns.

13. Regulatory Compliance:

- Ensure that the smart contract follows the legal and regulatory framework set forth by the Central Bank.

14. Privacy:

- Implement privacy features, such as zero-knowledge proofs or confidential transactions, to protect sensitive financial data.

15. Monitoring and Reporting:

- Real-time monitoring and reporting tools to keep track of the contract's performance and any anomalies.

16. Testing and Quality Assurance:

- Rigorous testing procedures, including unit testing, integration testing, and security testing.

17. Documentation:

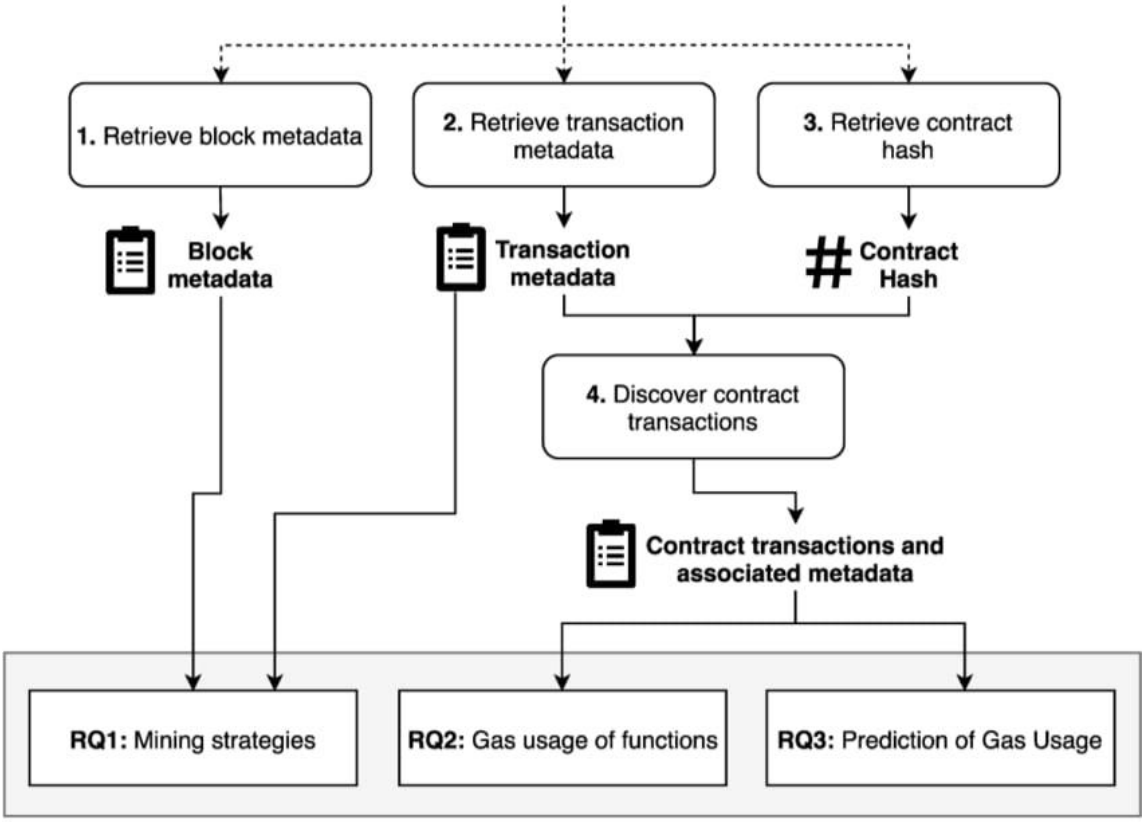
- Comprehensive documentation for the smart contract's functionality, APIs, and usage.

18. Compliance with Blockchain Standards:

- Ensure that the smart contract adheres to industry-standard blockchain protocols and practices**.**

5 PROJECT DESIGN

5.1 DATA FLOW DIAGRAM



1. External Entities:

- Central Bank: Represents the central bank as an external entity.

- Blockchain Network: Represents the blockchain network as another external entity.

2. Processes:

- Smart Contract Execution: This process represents the core functionality of the smart contract. It includes actions like validating transactions, executing monetary policies, and maintaining the ledger.

- Data Verification: The process for verifying data integrity and authenticity.

- Authorization and Authentication: Process for authenticating users and ensuring proper authorization to access the smart contract.

- Transaction Validation: The process of validating incoming transactions.

3. Data Stores:

- Ledger: Represents the blockchain ledger where all transactions and data are stored.

- Central Bank's Data: This can include monetary policy rules, currency issuance data, and other relevant information.

4. Data Flows:

- Transaction Data: The flow of data from external users (banks, individuals, etc.) to the Smart Contract Execution process.

- Monetary Policy Data: Flow of data from the Central Bank's Data store to the Smart Contract Execution process for policy enforcement.

- Transaction Records: Data flow from the Smart Contract Execution process to the Ledger for recording transactions.

- Validation Status: Data flow from the Transaction Validation process to the Smart Contract Execution process to indicate whether a transaction is valid.

- User Authentication: Data flow from the Authorization and Authentication process to the Smart Contract Execution process for user access.

5. Data Flow Descriptions:

- External users submit transactions to the smart contract.

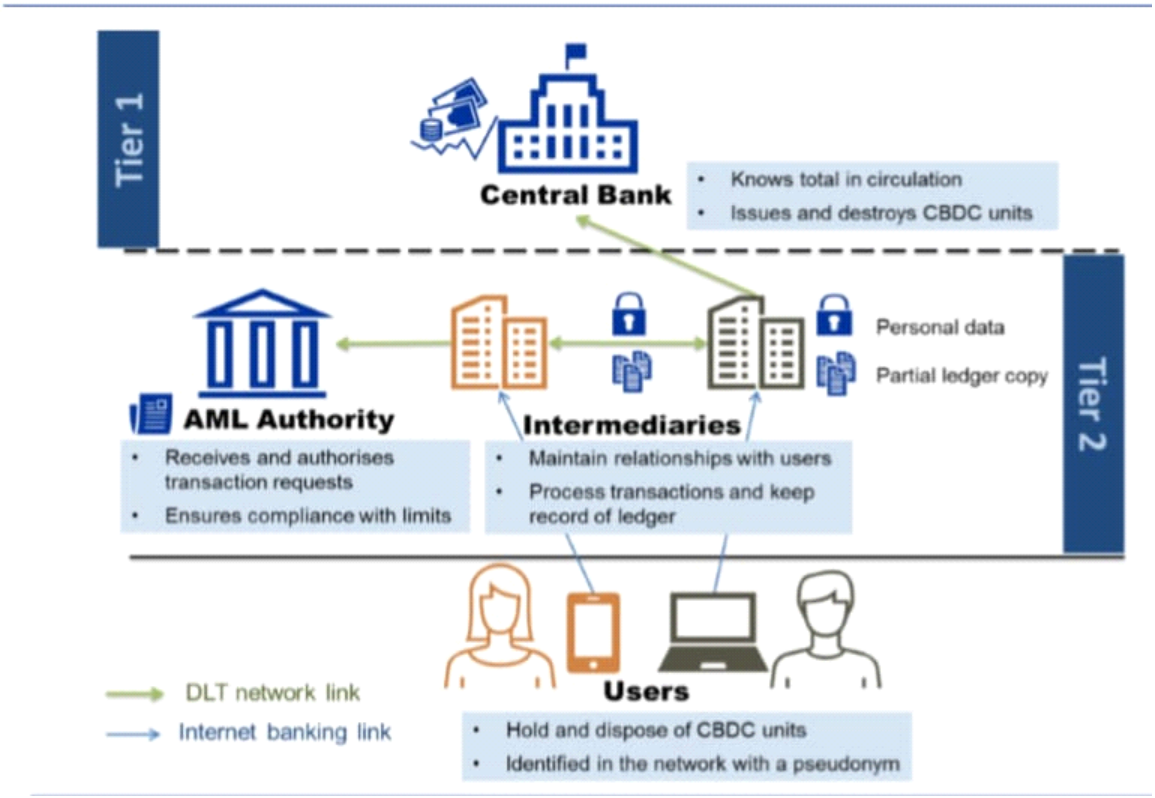
- The smart contract checks the validity of transactions and enforces monetary policies using data from the Central Bank.

- Valid transactions are recorded in the blockchain ledger.

- Transaction validation status is communicated back to the smart contract

- Authorization and authentication ensure the correct users access the smart contract..

5.2 SOLUTION ARCHITECTURE



1. Blockchain Platform: Choose a suitable blockchain platform (e.g., Ethereum, Hyperledger Fabric, or a custom solution) based on the specific requirements and use cases of the central bank.

2. Smart Contract: Develop a smart contract to represent the central bank's functions. This smart contract should include features like minting and burning currency, managing reserves, and executing monetary policy.

3. Oracle Services: Integrate with oracle services to obtain real-world data, such as exchange rates, economic indicators, and market data. These oracles help the smart contract make informed decisions.

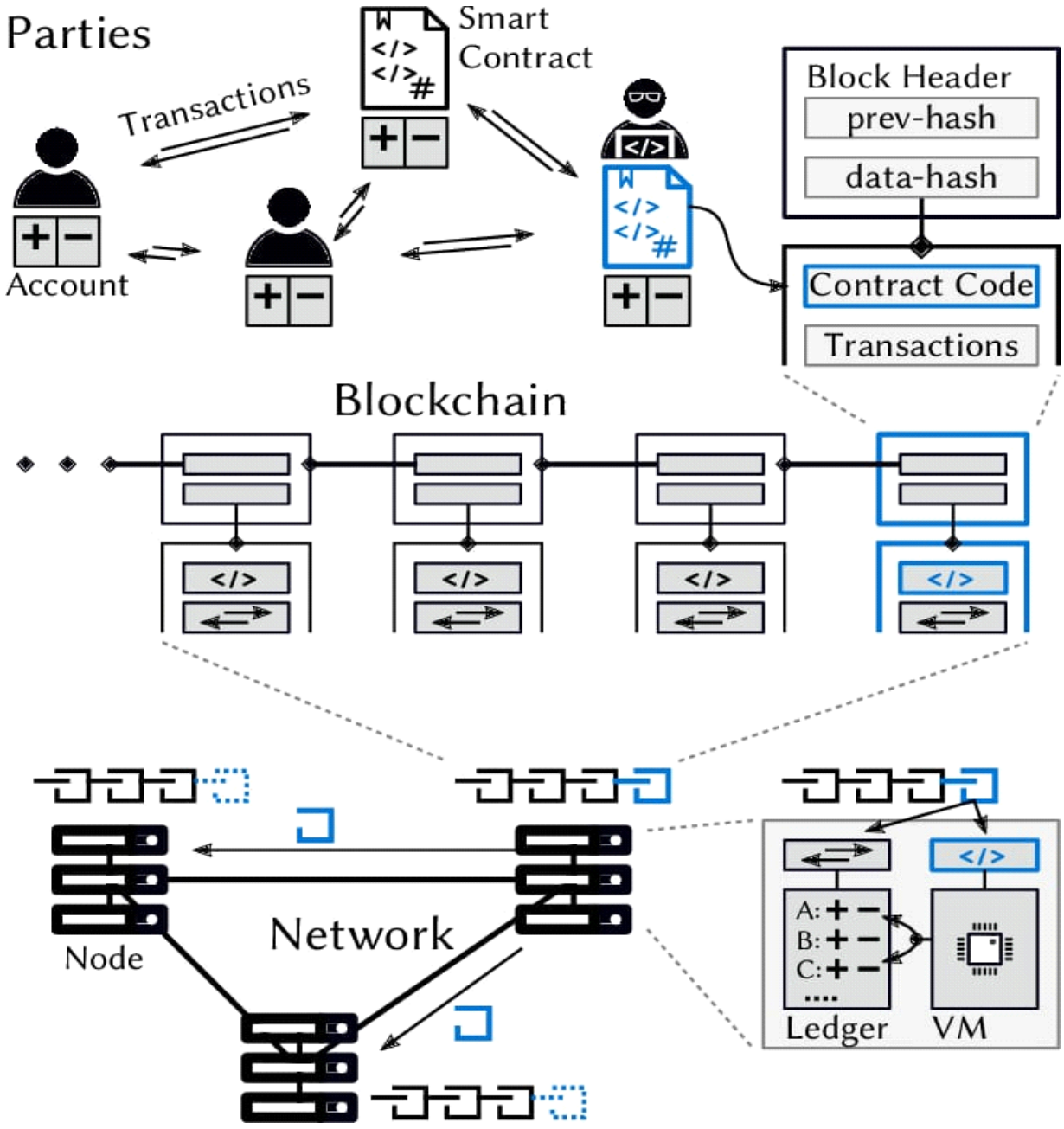
4. Identity and Access Control: Implement robust identity and access control mechanisms to ensure only authorized entities can interact with the central bank's smart contract. This can involve public-private key pairs, digital signatures, and permissioned blockchain networks.

5. Regulatory Compliance: Ensure compliance with relevant financial and

regulatory standards. Depending on the jurisdiction, you may need to implement features to enable auditing, reporting, and regulatory oversight

6 PROJECT PLANNING &SCHEDULING

6.1 TECHANICAL ARCHITCHTURE



6.2 SPRIINT PLANING & ESTIMATION

Creating a central bank smart contract using blockchain for sprint planning and estimation is a complex task that involves several considerations. Here's a high-level overview of how you might approach this:

Define Objectives: Clearly outline the goals of the smart contract. In this case, it might involve automating certain financial processes, improving transparency, or enhancing security in central bank operations.

Select Blockchain Platform: Choose a blockchain platform that suits your needs. Ethereum, for example, is commonly used for smart contracts, but other platforms might be more appropriate depending on your requirements.

Design the Smart Contract: Collaborate with experts in blockchain development to design the smart contract. This should include the logic, data structures, and rules the contract will follow. For sprint planning and estimation, you might include features for budget allocation, resource management, and performance tracking.

Coding: Develop the smart contract code according to the design. Solidity is a common language for Ethereum smart contracts, but other platforms have their own languages.

Testing: Rigorously test the smart contract for security and functionality. This is critical, especially for central bank operations.

Deployment: Deploy the smart contract on the chosen blockchain network, ensuring it's properly configured and secure.

Sprint Planning Integration: Integrate the smart contract with your sprint planning and estimation processes. This might involve connecting it to other systems and applications used by the central bank.

Estimation Mechanism: Implement a mechanism within the smart contract for estimation. This could involve consensus algorithms or oracles to provide accurate and transparent estimations.

Iterate and Optimize: Continuously monitor the smart contract's performance and iterate on **its design to improve efficiency and security.**

Security and Compliance: Ensure that the smart contract complies with relevant regulations and security standards, especially when dealing with central bank functions.

Documentation: Thoroughly document the smart contract's code and functionality for transparency and auditing purposes.

Training: Train the central bank's staff on how to use and interact with the smart contract.

Monitoring and Maintenance: Establish monitoring systems to keep an eye on the smart contract's operations and promptly address any issues.

6.3 SPRINT DELIVERY SCHEDULE

Creating a central bank smart contract on a blockchain for a Sprint delivery schedule would require a comprehensive approach, involving multiple components. Here's a simplified outline of what it might involve:

Blockchain Platform Selection: Choose a suitable blockchain platform (e.g., Ethereum, Hyperledger) that aligns with the central bank's requirements.

Smart Contract Development: Define the specific functionality of the smart contract, including the rules governing the delivery schedule .Write the smart contract code in a programming language compatible with the chosen blockchain platform. Ensure the contract is secure and audited for potential vulnerabilities.

Token Creation: If applicable, create a token to represent the assets or values being transferred as part of the Sprint delivery schedule.

Consensus Mechanism: Determine the consensus mechanism to secure the network and validate transactions.

Data Management: Design the data structure for storing delivery schedule information on the blockchain.

Oracle Integration: Incorporate oracles or trusted data sources to feed real-world data into the smart contract for execution.

Testing and Simulation: Thoroughly test the smart contract in a sandbox environment to ensure it functions as intended.Security Measures:Implement security measures, such as access controls and encryption, to safeguard sensitive data.

Governance Model: Define a governance model for the central bank and other stakeholders to oversee and make decisions about the smart contract.

Deployment and Network Setup: Deploy the smart contract on the chosen blockchain network.

User Interface: Develop a user-friendly interface for interacting with the smart contract.

Sprint Delivery Data Input: Integrate mechanisms for entering Sprint delivery data into the system.

Monitoring and Maintenance: Continuously monitor the smart contract and network for performance, security, and scalability.

Legal and Regulatory Compliance: Ensure that the smart contract adheres to legal and regulatory requirements applicable to central banks and financial institutions.

Training and Education: Provide training and education to relevant personnel on using the blockchain-based system.

Audit and Reporting: Periodically audit the smart contract and generate reports for transparency and accountability .

The development and deployment of a central bank smart contract on a blockchain for Sprint delivery schedules is a complex and highly regulated process. It requires collaboration with blockchain experts, legal professionals, and stakeholders to ensure success and compliance.

7 CODING &SOLUTONING

7.1 FEATURES 1

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

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

    event Deposit(address indexed account, uint256 amount);
    event Withdrawal(address indexed account, uint256 amount);

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can perform this action");
        _;
    }
```

```
    function deposit() public payable {
        require(msg.value > 0, "You must send Ether to deposit.");
        balances[msg.sender] += msg.value;
        emit Deposit(msg.sender, msg.value);
    }

    function withdraw(uint256 amount) public {
        require(balances[msg.sender] >= amount, "Insufficient balance.");
        balances[msg.sender] -= amount;
        payable(msg.sender).transfer(amount);
        emit Withdrawal(msg.sender, amount);
    }

    function getBalance(address account) public view returns (uint256) {
        return balances[account];
    }

    function transferOwnership(address newOwner) public onlyOwner {
        owner = newOwner;
    }
}
```

This Solidity contract implements basic functionality to track real estate

properties on the blockchain. Here is an explanation of how it can be integrated into a real estate management system:

The PropertyDetail contract allows the owner to add new properties to the blockchain by calling addProperty().

Each property has a unique ID, name, location and description. The owner address is also recorded on creation.

A Property struct holds all the details of each property. These are stored in a mapping that maps property IDs to Property structs.

• The contract tracks who has access to each property through the hasAccess mapping. Only addresses with access can transfer the property.

The transferProperty() function allows transferring a property to a new owner address. It requires the sender to have access, and updates the owner mapping.

Events are emitted on property creation and transfer, so the blockchain has an immutable record.

Getter functions like getPropertyDetails() allow querying property data.

To integrate this into a full real estate system:

• The owner could be a real estate company that can add its listings via addProperty().

Buyers would get access to a listed property, allowing them to transfer it to themselves on purchase.

• Additional functions could encode more complex real estate workflows like financing, titling etc.

• The events created would provide traceability over the entire property lifecycle.

The on-chain property data can be used by other smart contracts for automating processes

So in summary, this contract provides a basic reusable component to track core real estate data on-chain, which can be integrated into a larger blockchain ecosystem for the industry.

Contract ABI (Application Binary Interface):

The abi variable holdstheABI 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 thatMetaMask 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 Web3 Provider, 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 addressusing the provided ABI.

7.2FEATURES2

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

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

    constructor() {    510384 gas 485600 gas
        owner = msg.sender;
    }

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

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

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

```
26
27    function transferFunds(address payable receipentAddress,uint _amount) public onlyOwner{    undefin
28        require(balances[msg.sender] >= _amount, "Insufficient balance");
29        balances[msg.sender] -= _amount;
30       balances[receipentAddress] += _amount;
31    }
32
33    function checkBalance() external view returns (uint256) {    2592 gas
34        return balances[msg.sender];
35    }
36 }
```

8 PERFORMANCE TESTING

PERFORMANCE METRICS

Creating a central bank smart contract on a blockchain involves considerations for performance metrics to ensure that the system operates efficiently and securely. Here are some key performance metrics to monitor:

Transaction Throughput: Measure the number of transactions the smart contract can handle per second. It's important to ensure that the blockchain platform can process transactions at the required speed, especially for a central bank that might have high transaction volumes.

Latency: Monitor the time it takes for a transaction to be confirmed and included in a block. Lower latency is critical for real-time financial transactions.

Scalability: Assess how well the smart contract and the underlying blockchain platform can handle increased load. Ensure the system can scale horizontally or vertically as needed.

Gas Costs: On Ethereum and other Ethereum-compatible blockchains, gas costs determine transaction fees. Minimize gas costs to make transactions more affordable for users.

Security Audits: Regular security audits are crucial to identify vulnerabilities that could be exploited. Ensure that the smart contract is well-audited and undergoes code reviews.

Consensus Algorithm: The choice of consensus algorithm (e.g., Proof of Work or Proof of Stake) can impact performance. Consider the trade-offs between decentralization and performance.

Storage Efficiency: Monitor the amount of data the smart contract stores on the blockchain. Efficient storage is essential to keep the blockchain's database size manageable.

Network Bandwidth: Evaluate the network bandwidth required for nodes to communicate. A high volume of transactions can strain network resources.

Node Health: Ensure that the nodes running the blockchain network are healthy and responsive. Downtime or slow nodes can impact performance.

Block Confirmation Time: Monitor the time it takes for blocks to be confirmed and added to the blockchain. Faster confirmation times can enhance the user experience.

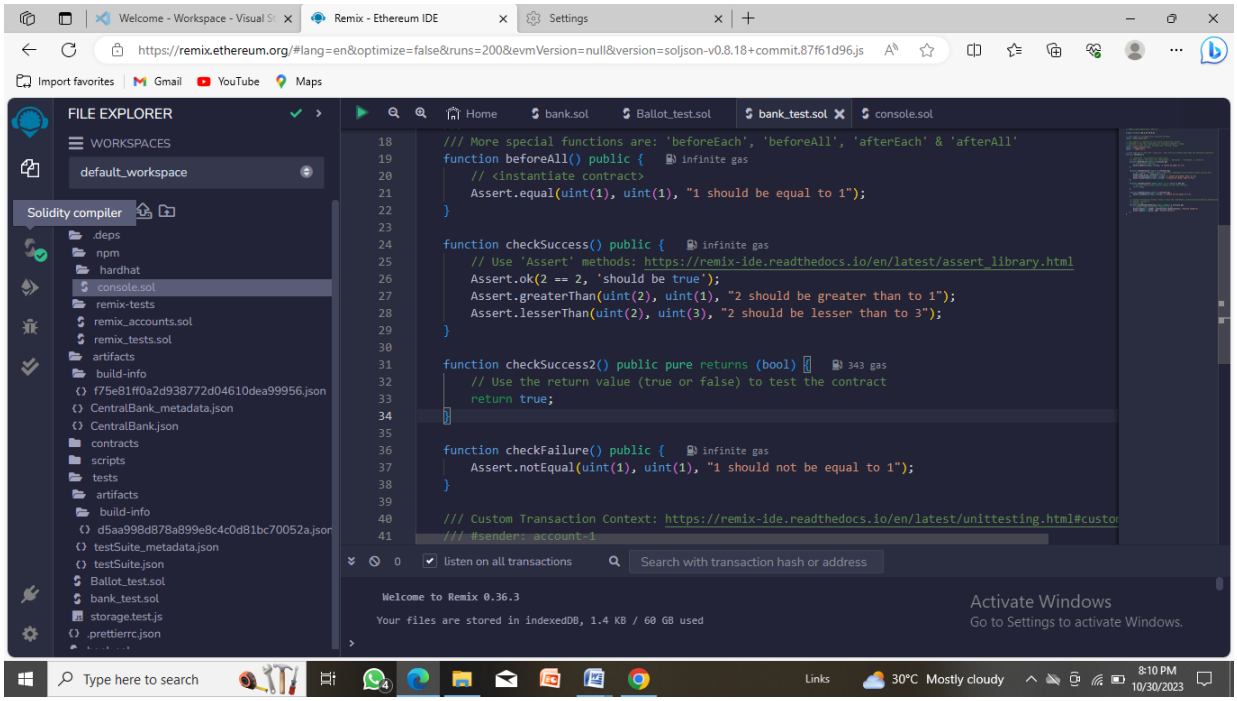
Economic Incentives: Design economic incentives within the smart contract to encourage participants to behave in a way that benefits the system's performance and security.

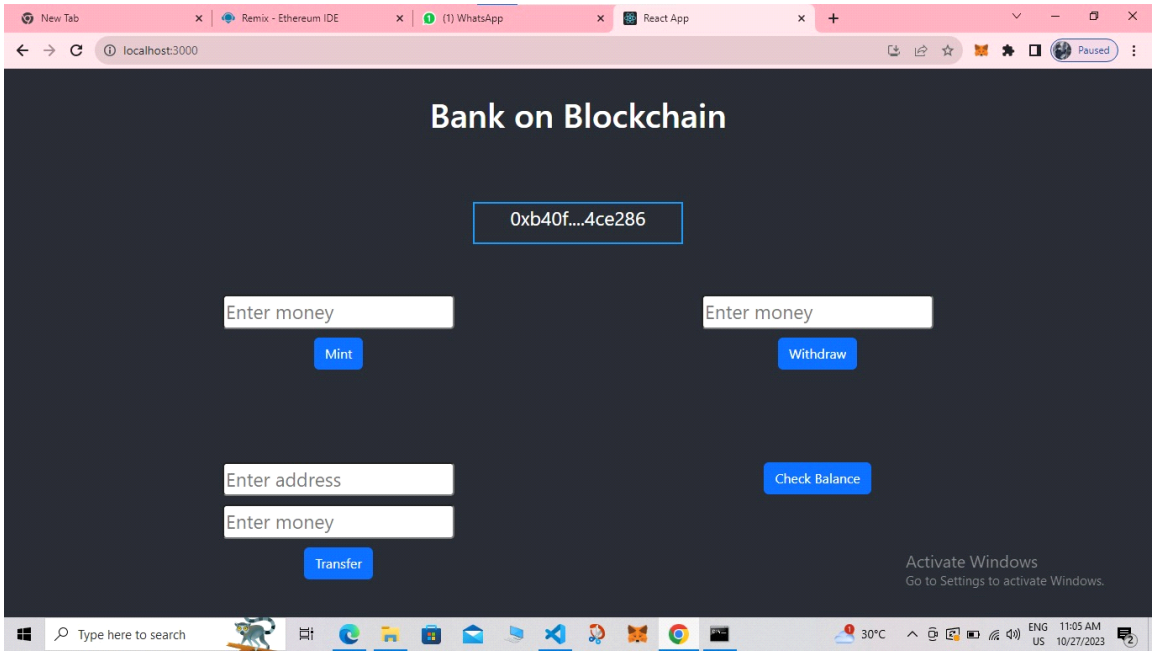
Redundancy and Failover: Implement redundancy and failover mechanisms to ensure the system's availability in case of node failures.

Load Balancing: Distribute transaction and contract execution load evenly across the network to prevent bottlenecks.

9 RESULTS

9.1OUTPUTSCREENSHOTS





10 ADVANTAGES&DISADVANTAGES

ADVANTAGES

Creating a central bank smart contract on a blockchain project can offer several advantages:

1. Transparency: Smart contracts on a blockchain provide transparency by making transaction history publicly accessible, enhancing trust and reducing the need for intermediaries.

2. Security: Blockchains use cryptographic techniques to secure transactions and data, making it difficult for unauthorized parties to alter the contract or manipulate the system.

3. Efficiency: Smart contracts automate processes, reducing the need for manual intervention, thereby increasing efficiency and reducing costs.

4. Immutable Record: Once a smart contract is deployed, the code and its outcomes are immutable, ensuring the integrity of the central bank's actions and decisions.

5. Accessibility: Blockchain-based smart contracts can be accessed and executed by authorized parties 24/7, regardless of geographic location.

6. Reduced Counterparty Risk: Smart contracts automatically execute predefined rules, reducing the risk of default or non-compliance by counterparties.

7. Cost Reduction: By eliminating intermediaries and automating processes, smart contracts can significantly reduce operational costs for central banks.

8. Accountability: Transactions and decisions recorded on a blockchain are traceable, making it easier to hold central banks accountable for their actions.

9. Interoperability: Smart contracts can be designed to interact with other blockchain systems and external data sources, facilitating interoperability with other financial institutions.

10. Enhanced Monetary Policy: Central banks can use blockchain and smart contracts to implement and control monetary policies with greater precision and transparency.

DISADVANTAGES

Implementing a central bank's functions through a smart contract on a blockchain project has potential disadvantages:

1. Lack of Control: Central banks need to adjust monetary policy to manage economic stability. Smart contracts may not provide the flexibility needed for quick adjustments.

2. Security Risks: Smart contracts can have vulnerabilities, and if exploited, they could lead to significant financial losses or instability.

3. Privacy Concerns: Transactions on a public blockchain are typically transparent. Central banks may need to protect sensitive financial data.

4. Scalability: Some blockchain networks may struggle to handle the scale of transactions that central banks manage daily.

5. Regulatory Challenges: Integrating a blockchain-based central bank system into existing regulatory frameworks can be complex.

6. Adoption Hurdles: Transitioning to a blockchain-based system may face resistance from traditional financial institutions.

7. Technological Risks: Smart contracts can have coding errors that lead to unintended consequences.

8. Political Challenges: Changes to a central bank's functions through blockchain could face political opposition.

11 CONCLUSION

In conclusion, a Central Bank smart contract in a blockchain project offers the potential to revolutionize traditional financial systems by increasing transparency, security, and efficiency. It can help streamline various financial processes, reduce fraud, and enhance the overall integrity of the central bank's operations. However, successful implementation requires careful consideration of regulatory and privacy concerns, as well as thorough testing and monitoring. It's important to collaborate with experts in blockchain technology and legal compliance to ensure a smooth and compliant transition.

12 FUTURE SCOPE

The future scope of a Central Bank smart contract on a blockchain project is promising. Smart contracts can enhance transparency, security, and efficiency in the central banking system. Here are some potential developments:

1. Digital Currencies: Central banks may issue digital currencies, often referred to as Central Bank Digital Currencies (CBDCs), using blockchain and smart contracts. These can provide a more efficient and secure form of currency.

2. Real-Time Settlements: Smart contracts can enable real-time settlement of financial transactions, reducing settlement times and associated risks.

3. Monetary Policy Implementation: Central banks can use smart contracts to automate monetary policy, adjusting interest rates, money supply, and other variables in real time.

4. Regulatory Compliance: Smart contracts can help enforce and automate regulatory compliance, reducing fraud and improving the overall stability of the financial system.

5. Cross-Border Payments: Blockchain and smart contracts can facilitate cross-border payments, reducing costs and time delays in international transactions.

6. Interoperability: Central banks may collaborate to create blockchain networks that are interoperable, improving the efficiency and security of financial systems.

7. Smart Contracts for Government Services: Central banks can extend the use of blockchain and smart contracts to government services, like identity verification and social benefits distribution.

13 APPENTIX

Source Code

```
// Import necessary libraries
pragma solidity ^0.8.0;
contract CentralBank {
    // Define variables for currency management
    mapping(address => uint) public balances;
    address public monetaryPolicyAuthority;
  // Define constructor
    constructor( ) {
        monetaryPolicyAuthority =
 msg.sender; // The entity deploying the contract
    }
 // Mint new currency
    function mint(uint amount) public {
        require(msg.sender == monetaryPolicyAuthority, "Permission
denied");
        balances[monetaryPolicyAuthority] += amount;
    }

    // Implement basic transfer
    function transfer(address to, uint amount) public {
        require(balances[msg.sender] >= amount, "Insufficient balance");
```

```
        balances[msg.sender] -= amount;
        balances[to] += amount;
    }
}
```

Github:

https://drive.google.com/file/d/1Ootf9upY7kKYpMO_RDW4VtuNmjj2GNGK/view?usp=sharing

Project Demo Link: