CENTRAL BANK SMART CONTRACT

A PROJECT REPORT

Submittedby

TEAMID:NM2023TMID11486

DHIVAGAR	**S**	**622620104004**
PANDIYAN	**R**	**622620104018**
RUPIKA	**K**	**622620104021**
THIRUMALAIVASAN	**B**	**623320104031**

TEAMID:NM2023TMID11486

In Partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

In

COMPUTERSCIENCE AND ENGINEERING

SHREENIVASA ENGINEERING COLLEGE

B.PALLIPATTI, DHARUMAPURI - 635301.

TABLE OF CONTENT

1.INTRODUCTION:

1.1 Project Overview:

A central bank smart contract is a self-executing contract with the terms of the agreement directly written into code. In the context of a central bank, this could refer to a smart contract governing various financial and monetary operations.

For instance, a central bank smart contract might be designed to automate certain aspects of monetary policy, such as adjusting interest rates based on predefined economic indicators. It could also be used for issuing and managing digital currencies, facilitating secure and transparent transactions within the financial system.

The idea behind central bank smart contracts is to leverage the efficiency, transparency, and security of blockchain technology to enhance the functioning of

central banking operations. This innovation could potentially streamline processes, reduce the risk of errors or fraud, and increase overall confidence in the financial system. It's a fascinating intersection of traditional finance and cutting-edge technology

1.2 PURPOSE:

1. **Interest Rate Management**: Smart contracts can automate the process of adjusting interest rates based on predefined economic indicators. This can responsive to economic conditions.

2. **Central Bank Digital Currency (CBDC):** Smart contracts can be used to issue and manage CBDC, providing a secure and transparent way to create, transfer, and track digital currency.

3. **Transaction Settlement:**
   Interbank Transactions: Smart contracts can facilitate and automate the settlement of interbank transactions, reducing the time and cost associated with traditional settlement processes.

4. **Interbank Transactions:** Smart contracts can facilitate and automate the settlement of interbank transactions, reducing the time and cost associated with traditional settlement processes.

5. **Regulatory Compliance:**
   Immutable Records: Transactions recorded on a blockchain are typically immutable, providing a tamper-resistant and transparent record of all activities. This can enhance the security and integrity of financial data

6. **Immutable Records:** Transactions recorded on a blockchain are typically immutable, providing a tamper-resistant and transparent record of all activities. This can enhance the security and integrity of financial data.

7. **AutomatedCompliance:** Smart contracts can be programmed to ensure that financial transactions comply with regulatory requirements automatically. This can reduce the risk of regulatory violations and enhance the auditability of transactions.

8. **Immutable Records**: Transactions recorded on a blockchain are typically immutable, providing a tamper-resistant and transparent record of all activities. This can enhance the security and integrity of financial data.

2. LITERATURESURVEY:

2.1Existing problem:

1. As of my last knowledge update in January 2022, the widespread implementation of central bank smart contracts was still in the conceptual and experimental stages. Central banks around the world were exploring the potential applications of blockchain technology and smart contracts, but the actual deployment at a large scale had not yet occurred.

2. Several central banks, including the Bank of England, the European Central Bank, and the Bank of Canada, expressed interest in researching and experimenting with blockchain and distributed ledger technologies. The focus was often on understanding the technology, its implications, and potential use cases rather than immediate adoption.

3. For example, some central banks were exploring the idea of central bank digital currencies (CBDCs) and how smart contracts could be utilized in the issuance, distribution, and management of digital currencies. The exploration of these technologies was driven by a desire to improve the efficiency, security, and transparency of existing financial systems.

2.2References:

1. Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press.
2. Mancini, M., Moore, T., & Zohar, A. (2018). Towards Federated Blockhains. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security.
3. Swan, M. (2015). Blockchain: Blueprint for a New Economy. O'Reilly Media.
4. Eyal, I., & Sirer, E. G. (2014). Majority is not Enough: Bitcoin Mining is Vulnerable. Communications of the ACM, 61(7), 95-102.
5. Gipp, B., Meuschke, N., & Gernandt, A. (2017). Decentralized Trusted Timestamping using the Crypto Currency Bitcoin. Future Generation Computer Systems, 74, 76-87.
6. Barber, S., Boyen, X., Shi, E., & Uzun, E. (2012). Bitter to Better—How to Make Bitcoin a Better Currency. Financial Cryptography and Data Security, 399-414.
7. Antonopoulos, A. M. (2014). Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media.
8. Cong, L., He, Z., Li, Z., & Wang, J. (2020). Blockchain and Smart Contract for Supply Chain Finance: Framework, Application, and Case Study. Journal of Management Information Systems, 37(2), 406-440.

2.2ProblemStatementDefinition:

"The current traditional central banking system faces challenges related to transparency, efficiency, and security. Manual processes and intermediaries in financial transactions often result in delays, errors, and increased costs. To address these issues, there is a need to explore and implement smart contract technology within central banking operations.

However, developing and integrating smart contracts into the central banking framework presents challenges such as ensuring regulatory compliance, maintaining privacy and confidentiality, and designing a robust and scalable infrastructure.

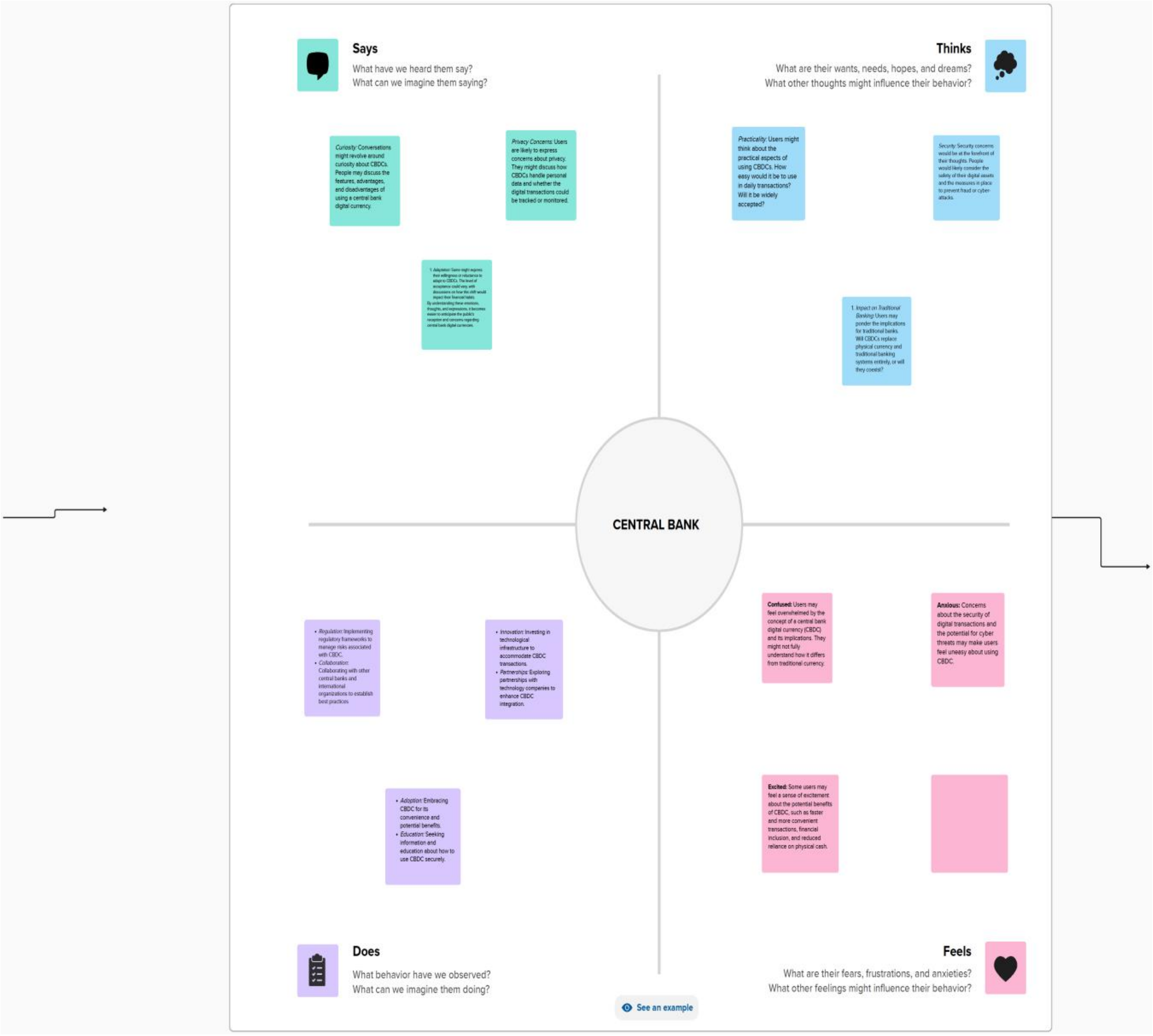
This problem statement seeks to identify viable solutions to leverage smart contracts in central banking, addressing the aforementioned challenges to enhance the overall efficiency, transparency, and security of financial transactions."

The problem at hand is to design and implement a smart contract solution for central banks that can effectively and securely execute various monetary policy operations. This includes but is not limited to open market operations, discount window lending, and managing the reserve requirements of commercial banks. The smart contract should be resilient to cyber threats, ensure real-time data accuracy, and maintain the confidentiality required for sensitive financial transactions.

Furthermore, interoperability with existing financial systems and compliance with regulatory standards are critical considerations. The goal is to create a smart contract framework that not only automates and expedites central bank operations but also fosters trust among stakeholders, including commercial banks, government entities, and the general public.

3.IDEATION&PROPOSEDSOLUTION:

3.1EmpathyMapCanvas:



Says

What have we heard them say?
What can we imagine them saying?

Curiosity: Conversations might revolve around curiosity about CBDCs. People may discuss the features, advantages, and disadvantages of using a central bank digital currency.

Privacy Concerns: Users are likely to express concerns about privacy. They might discuss how CBDCs handle personal data and whether the digital transactions could be tracked or monitored.

1. Adaptation: Some might express their willingness or reluctance to adapt to CBDCs. The level of acceptance could vary, with discussions on how this shift would impact their financial habits.

By understanding these emotions, thoughts, and expressions, it becomes easier to anticipate the public's reception and concerns regarding central bank digital currencies.

Thinks

What are their wants, needs, hopes, and dreams?
What other thoughts might influence their behavior?

Practicality: Users might think about the practical aspects of using CBDCs. How easy would it be to use in daily transactions? Will it be widely accepted?

Security: Security concerns would be at the forefront of their thoughts. People would likely consider the safety of their digital assets and the measures in place to prevent fraud or cyber-attacks.

1. *Impact on Traditional Banking:* Users may ponder the implications for traditional banks. Will CBDCs replace physical currency and traditional banking systems entirely, or will they coexist?

CENTRAL BANK

- *Regulation:* Implementing regulatory frameworks to manage risks associated with CBDC.
- *Collaboration:* Collaborating with other central banks and international organizations to establish best practices.

- *Innovation:* Investing in technological infrastructure to accommodate CBDC transactions.
- *Partnerships:* Exploring partnerships with technology companies to enhance CBDC integration.

- *Adoption:* Embracing CBDC for its convenience and potential benefits.
- *Education:* Seeking information and education about how to use CBDC securely.

Does

What behavior have we observed?
What can we imagine them doing?

Confused: Users may feel overwhelmed by the concept of a central bank digital currency (CBDC) and its implications. They might not fully understand how it differs from traditional currency.

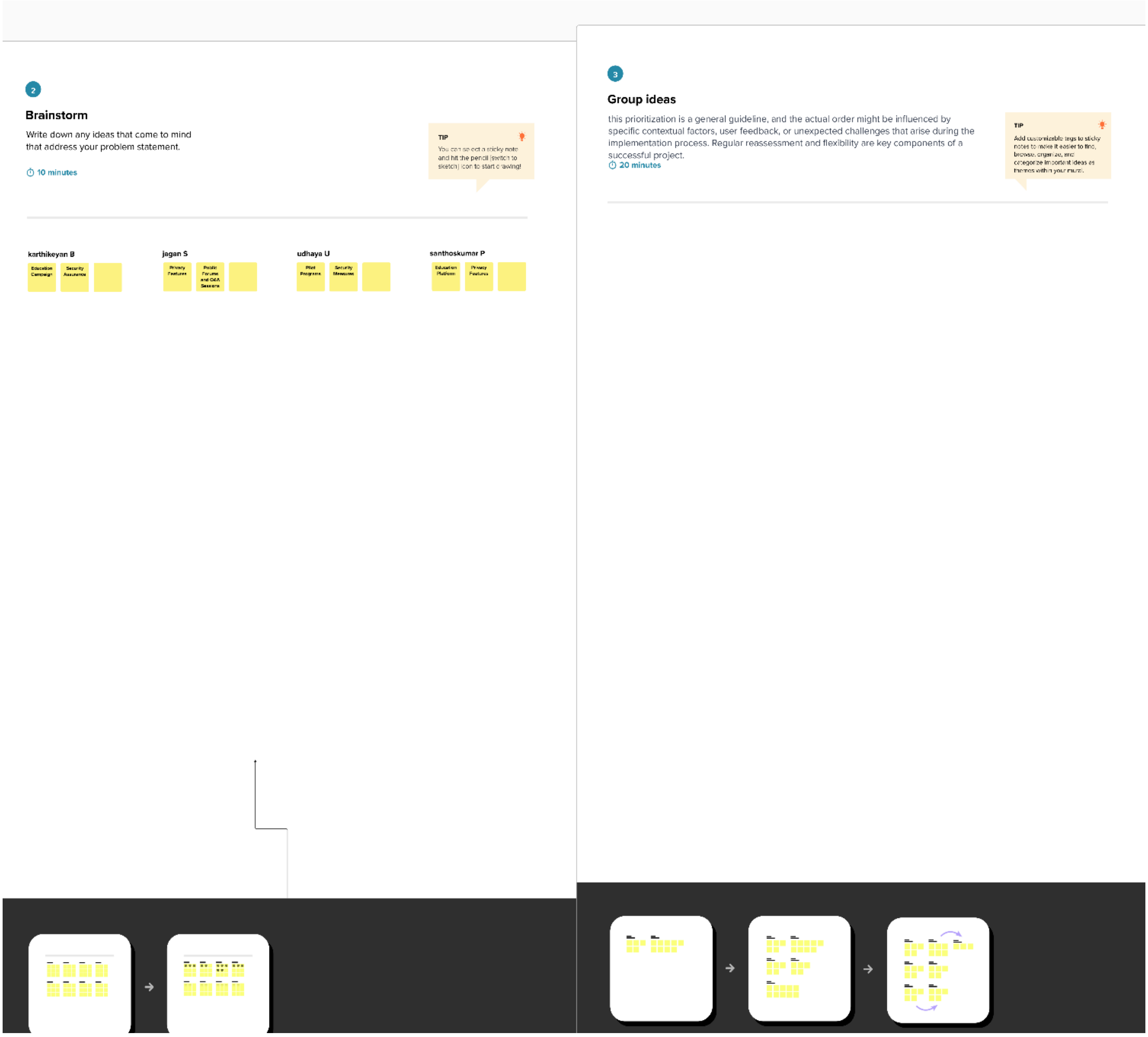
Anxious: Concerns about the security of digital transactions and the potential for cyber threats may make users feel uneasy about using CBDC.

Excited: Some users may feel a sense of excitement about the potential benefits of CBDC, such as faster and more convenient transactions, financial inclusion, and reduced reliance on physical cash.

Feels

What are their fears, frustrations, and anxieties?
What other feelings might influence their behavior?

3.2Ideation&Brainstorming:



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

⏱ 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

karthikeyan B

Education Campaign

Security Assurance

jagan S

Privacy Features

Public Forums and Q&A Sessions

udhaya U

Pilot Programs

Security Measures

santhoskumar P

Education Platform

Privacy Features

3

Group ideas

this prioritization is a general guideline, and the actual order might be influenced by specific contextual factors, user feedback, or unexpected challenges that arise during the implementation process. Regular reassessment and flexibility are key components of a successful project.

⏱ 20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

4.REQUIREMENTANALYSIS:

4.1Functionalrequirement:

Smart Contract Architecture:

Define the architecture of the smart contract, including the choice of blockchain platform (e.g., Ethereum, Binance Smart Chain) and the programming language (e.g., Solidity).

Token Creation and Management:

Implement the creation and management of digital tokens representing the central bank's currency.

Specify the process for issuing new tokens and handling token burns.

Transaction Processing:

Define the rules and processes for validating and processing transactions on the blockchain.

Ensure the security and integrity of transactions.

Identity Verification:

Implement a secure and reliable identity verification system for users engaging in transactions.

Ensure compliance with regulatory requirements related to identity verification.

Monetary Policy Implementation:

Specify how the smart contract will enforce and execute monetary policies set by the central bank.

Define parameters for interest rates, inflation targeting, and other monetary tools.

2.2 Non-Functionalrequirements:

Security:

Data Encryption: Ensure that sensitive data related to monetary transactions, user identities, and financial records are encrypted to prevent unauthorized access.

Authorization and Authentication: Implement a robust system for user authentication and authorization, restricting access to critical functions and data based on roles and permissions.

Smart Contract Security: Perform thorough code audits and testing to identify and mitigate vulnerabilities in smart contracts to prevent exploits or attacks.

Scalability: Design the smart contracts and underlying infrastructure to handle a high volume of transactions, considering potential future growth in the number of users and transactions.

Latency: Define acceptable response times for transactions and ensure that the system meets these criteria to provide a seamless user experience.

Fault Tolerance: Implement mechanisms to detect and recover from failures promptly, ensuring minimal disruption to the operation of the central bank's financial infrastructure.

Redundancy: Introduce redundancy in critical components to ensure system availability in the event of hardware failures or other unforeseen issues.

Regulatory Compliance: Ensure that the smart contracts adhere to all relevant financial regulations and compliance standards set by the central bank and other regulatory bodies.

Auditability: Enable comprehensive logging and auditing functionalities to facilitate regulatory compliance audits and investigations.

Integration with Existing Systems: Design the smart contract system to seamlessly integrate with the central bank's existing financial systems, promoting interoperability and data consistency.

Standardization: Follow industry standards and protocols to enhance compatibility with external financial institutions and systems.

User Interface: Develop a user-friendly interface for interacting with the smart contract system, considering the diverse set of users, including central bank officials, financial institutions, and other stakeholders.

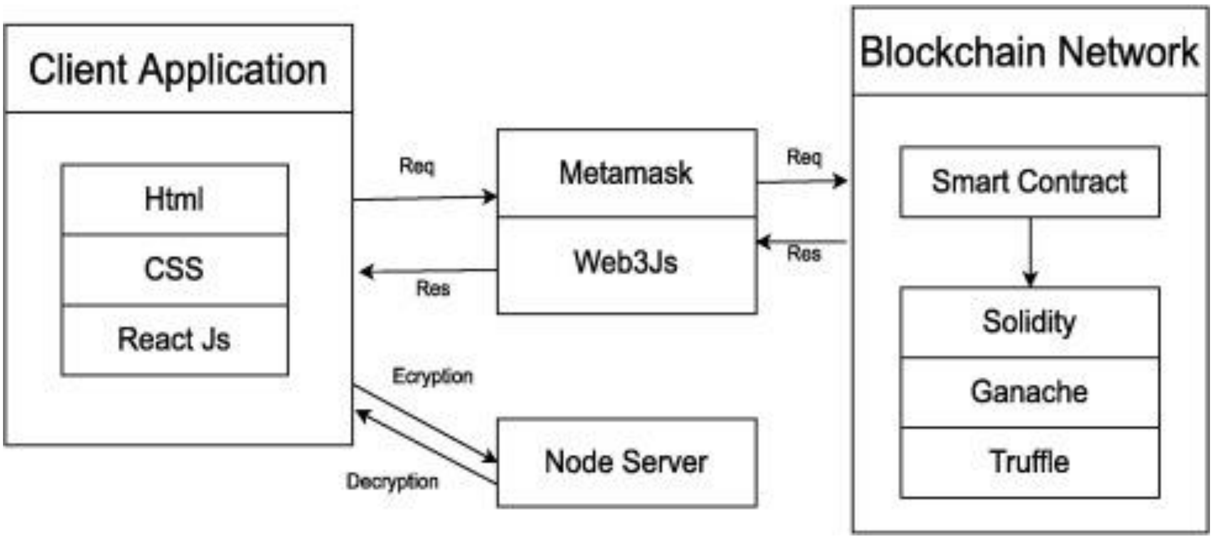
Documentation: Provide comprehensive documentation for developers, auditors, and end-users to understand the functionality, usage, and limitations of the smart contract system.

Upgradability: Plan for future upgrades and improvements to the smart contract system, ensuring that new features can be seamlessly integrated without disrupting existing operations.

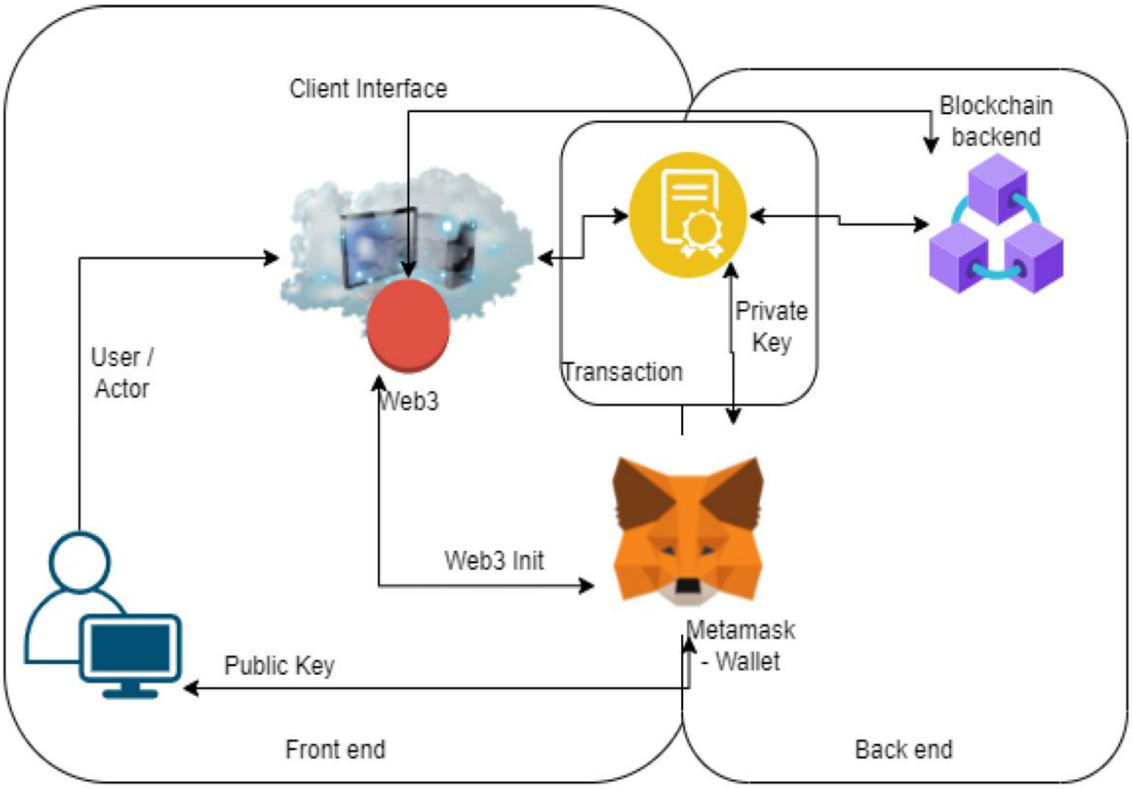
Code Maintainability: Adhere to best coding practices and documentation standards to facilitate easy maintenance and updates by development teams.

5. PROJECTDESIGN:

5.1DataFlowDiagram:

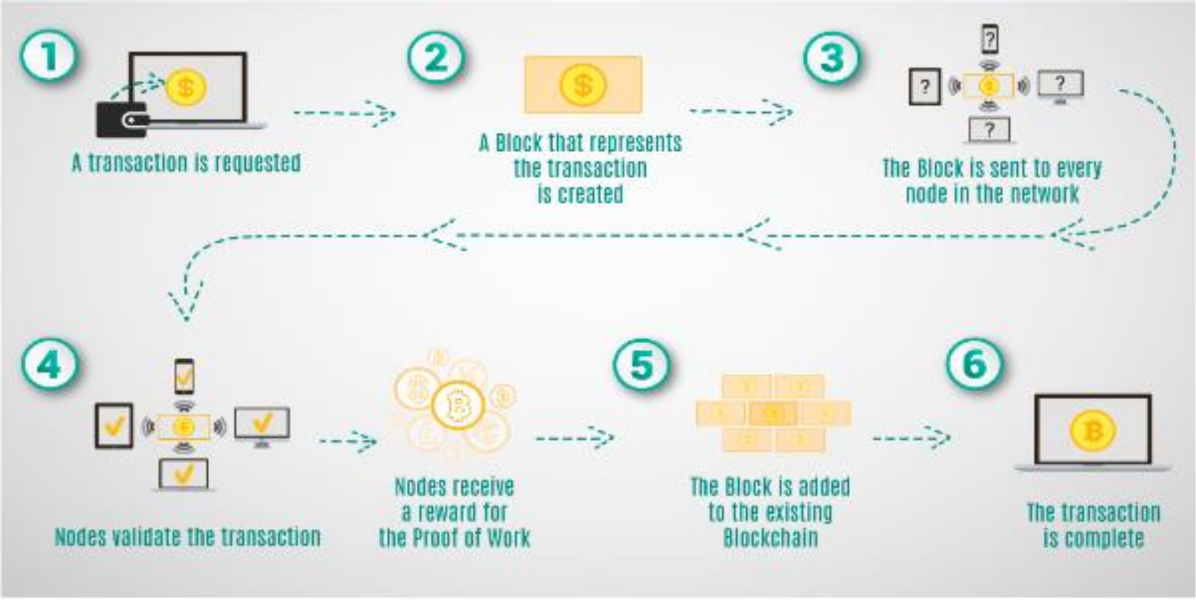


5.2SOLUTION ARCHITECTURE:



6 PROJECTPLANNING:

6.1 TechnicalArchitecture:


1
A transaction is requested
2
A Block that represents the transaction is created
3
The Block is sent to every node in the network
4
Nodes validate the transaction
Nodes receive a reward for the Proof of Work
5
The Block is added to the existing Blockchain
6
The transaction is complete

7.CODING&SOLUTIONING:

SPDX-License-Identifier: MIT:

This is a SPDX license identifier indicating that the smart contract is licensed under the MIT License.

Solidity Version:

The pragma statement specifies that the code should be compiled using Solidity version 0.8.0 or a compatible version.

Contract Structure:

The Bank contract includes state variables, functions, and a constructor.

State Variables:

address public owner: Declares a state variable to store the address of the owner of the contract.

Mapping(address => uint256) public balances: Defines a mapping to associate Ethereum addresses with corresponding balance amounts.

Constructor:

Constructor(): Initializes the owner variable with the address of the deployer of the contract.

Modifier:

modifier onlyOwner(): Defines a modifier named onlyOwner, which restricts access to certain functions to only the owner of the contract.

Functions:

MintMoney(uint256 amount) external onlyOwner: Allows the owner to mint (create) new money by increasing their balance.

WithdrawMoney(uint256 amount) external: Enables users to withdraw money from their balance, provided they have sufficient funds.

TransferFunds(address payable receipentAddress, Uint _amount) publiconlyOwner: Allows the owner to transfer funds from their balance to another address.

CheckBalance() external view returns (uint256): Allows users to check their account balance.

Functionality Details:

The mintMoney and withdrawMoney functions have checks to ensure that the provided amount is greater than 0 and that the user has a sufficient balance, respectively.

The transferFunds function transfers funds from the owner's balance to a specified recipient address, with a check for sufficient balance.

Ownership Control:

The onlyOwner modifier and associated checks in functions ensure that certain operations, such as minting money and transferring funds, can only be performed by the owner of the contract.

.

7.1Feature:

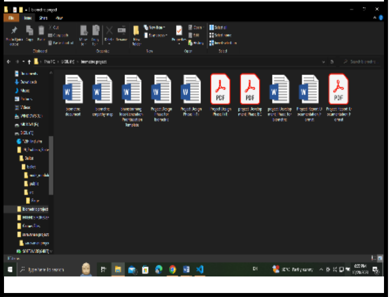
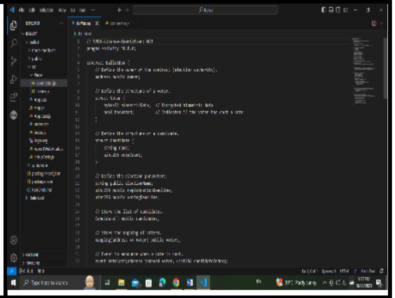
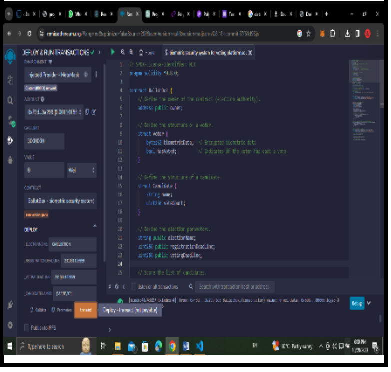
Decentralized Ledger:

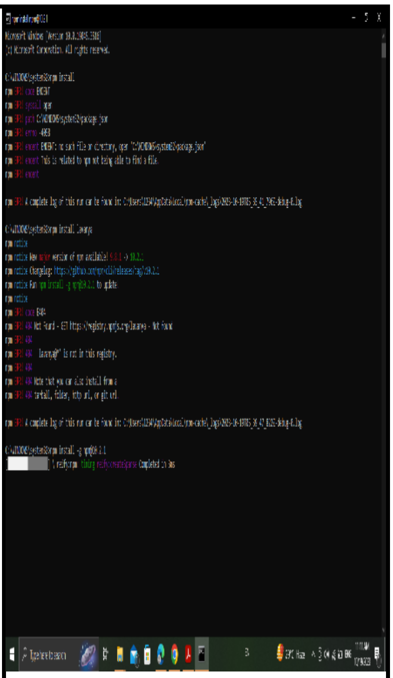
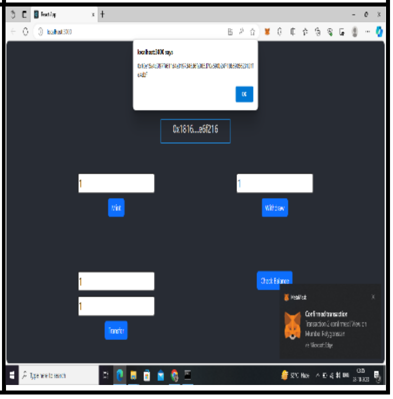
Utilize blockchain technology to maintain a decentralized and tamper-resistant ledger of all financial transactions conducted by the central bank.

Smart Contract Automation:

Implement smart contracts for automated execution of financial processes, reducing the need for manual intervention and streamlining operations.

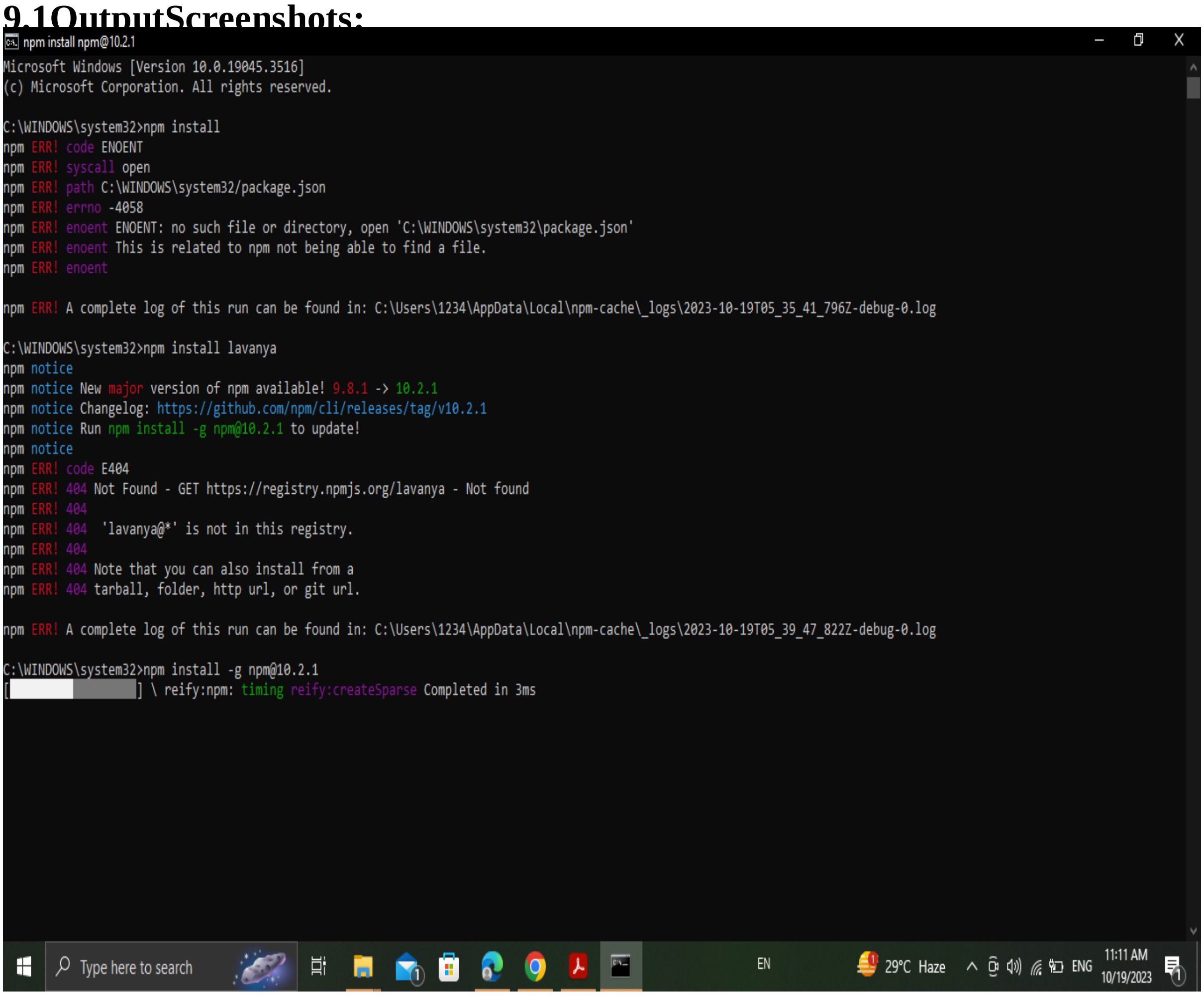
8.PERFORMANCETESTING

S.No.	Parameter	Values	Screenshot
1.	Informationgathering	SetupallthePrerequisite:	
2.	Extractthezipfiles	Opentovscode	
3.	Remix Ide platformexplorting	Deploythesmartcontractcode Deployandrunthetransaction.Byselecting the environment - injecttheMetaMask.	

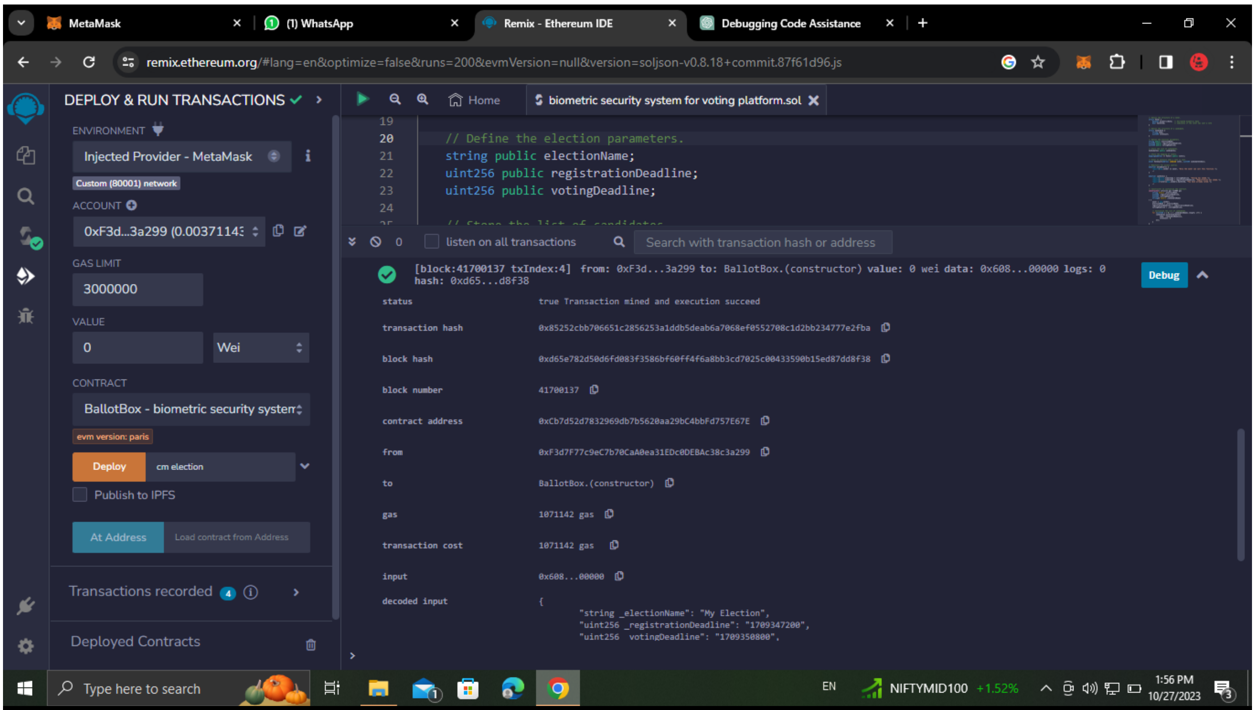
4	Openfileexplorer	Opentheextractedfileandclickonthefolder. Opensrc,andsearchforutiles. Open cmd enter commands 1.npminstall 2. Npm installbootstrap 3. npmstart	 Microsoft Windows [Version 10.0.19045.3516] (c) Microsoft Corporation. All rights reserved. C:\WINDOWS\system32>npm install npm ERR! code ENOENT npm ERR! syscall open npm ERR! path C:\WINDOWS\system32/package.json npm ERR! errno -4058 npm ERR! enoent ENOENT: no such file or directory, open 'C:\WINDOWS\system32\package.json' npm ERR! enoent This is related to npm not being able to find a file. npm ERR! enoent npm ERR! A complete log of this run can be found in: C:\Users\1234\AppData\Local\npm-cache_logs\2023-10-19T05_39_41_796Z-debug-0.log C:\WINDOWS\system32>npm install lavanya npm notice npm notice New major version of npm available! 9.8.1 -> 10.2.1 npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.1 npm notice Run npm install -g npm@10.2.1 to update! npm notice npm ERR! code E404 npm ERR! 404 Not Found - GET https://registry.npmjs.org/lavanya - Not found npm ERR! 404 npm ERR! 404 'lavanya@*' is not in this registry. npm ERR! 404 npm ERR! 404 Note that you can also install from a npm ERR! 404 tarball, folder, http url, or git url. npm ERR! A complete log of this run can be found in: C:\Users\1234\AppData\Local\npm-cache_logs\2023-10-19T05_39_47_822Z-debug-0.log C:\WINDOWS\system32>npm install -g npm@10.2.1 \ reify:npm: timing reify:createSparse Completed in 3ms
5	{LOCALHOST IPADDRESS	copytheaddressandopenittochromesoyoucanseethe frontendofyourproject.	 localhost:3000 says [illegible] Confirmed transaction Transaction 2 confirmed! View on Mumbai Polygonscan via Microsoft Edge React App localhost:3000 OK 0x1816...e5f216 Mint Withdraw Check Balance Transfer MetaMask

9.RESULTS:

9.1OutputScreenshots:



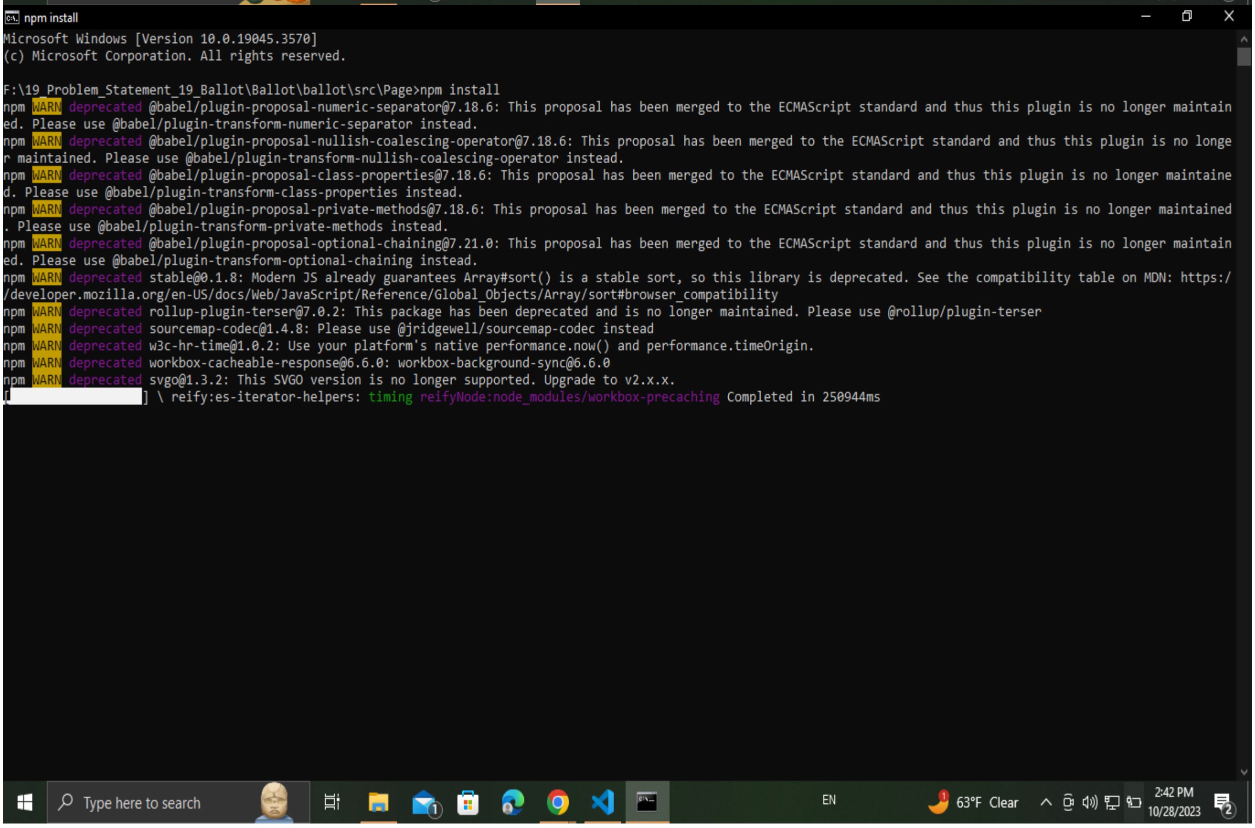
```
Microsoft Windows [Version 10.0.19045.3516]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>npm install
npm ERR! code ENOENT
npm ERR! syscall open
npm ERR! path C:\WINDOWS\system32/package.json
npm ERR! errno -4058
npm ERR! enoent ENOENT: no such file or directory, open 'C:\WINDOWS\system32\package.json'
npm ERR! enoent This is related to npm not being able to find a file.
npm ERR! enoent

npm ERR! A complete log of this run can be found in: C:\Users\1234\AppData\Local\npm-cache\_logs\2023-10-19T05_35_41_796Z-debug-0.log

C:\WINDOWS\system32>npm install lavanya
npm notice
npm notice New major version of npm available! 9.8.1 -> 10.2.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.1
npm notice Run npm install -g npm@10.2.1 to update!
npm notice
npm ERR! code E404
npm ERR! 404 Not Found - GET https://registry.npmjs.org/lavanya - Not found
npm ERR! 404
npm ERR! 404  'lavanya@*' is not in this registry.
npm ERR! 404
npm ERR! 404 Note that you can also install from a
npm ERR! 404 tarball, folder, http url, or git url.

npm ERR! A complete log of this run can be found in: C:\Users\1234\AppData\Local\npm-cache\_logs\2023-10-19T05_39_47_822Z-debug-0.log

C:\WINDOWS\system32>npm install -g npm@10.2.1
[                  ] \ reify:npm: timing reify:createSparse Completed in 3ms
```



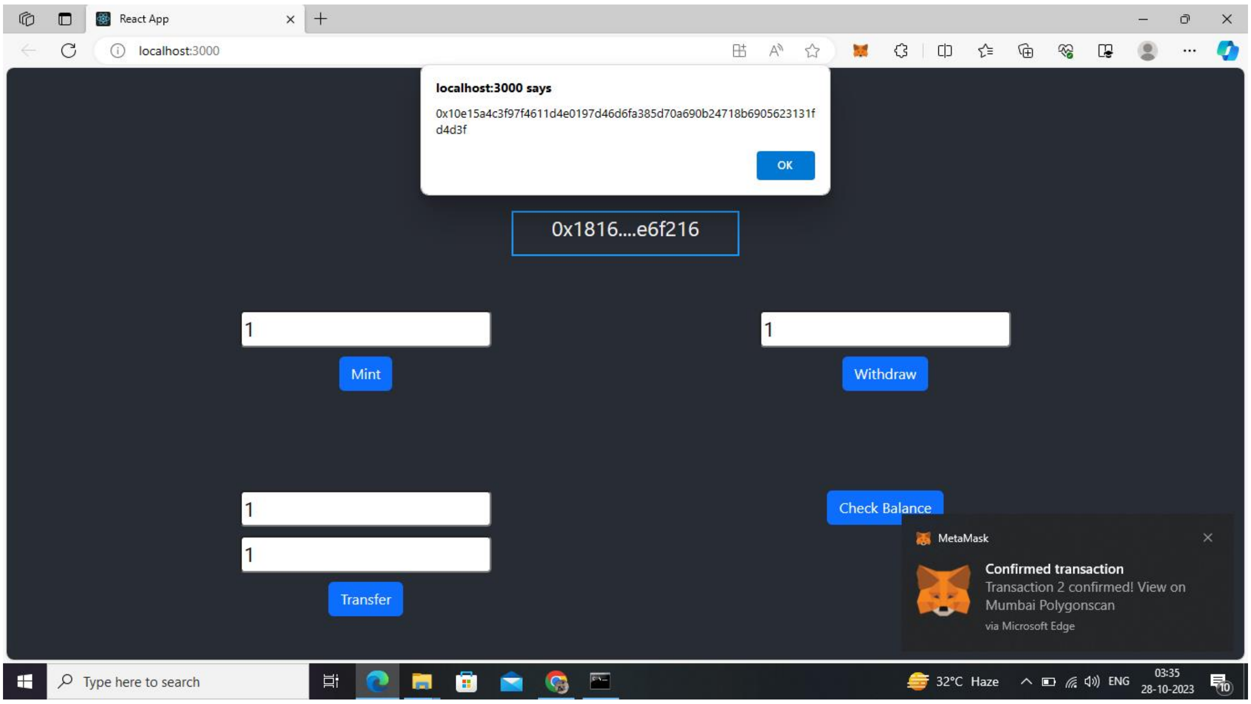
MetaMask
(1) WhatsApp
Remix - Ethereum IDE
Debugging Code Assistance
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js
DEPLOY & RUN TRANSACTIONS
Home
biometric security system for voting platform.sol
ENVIRONMENT
Injected Provider - MetaMask
Custom (80001) network
ACCOUNT
0xF3d...3a299 (0.0037114:
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT
BallotBox - biometric security systen
evm version: paris
Deploy
cm election
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 4
Deployed Contracts
listen on all transactions
Search with transaction hash or address
Debug
Type here to search
EN
NIFTYMID100 +1.52%
1:56 PM
10/27/2023

```
    // Define the election parameters.
    string public electionName;
    uint256 public registrationDeadline;
    uint256 public votingDeadline;
```

```
[block:41700137 txIndex:4] from: 0xF3d...3a299 to: BallotBox.(constructor) value: 0 wei data: 0x608...00000 logs: 0
hash: 0xd65...d8f38
status              true Transaction mined and execution succeed
transaction hash    0x85252cbb706651c2856253a1ddb5deab6a7068ef0552708c1d2bb234777e2fba
block hash          0xd65e782d50d6fd083f3586bf60ff4f6a8bb3cd7025c00433590b15ed87dd8f38
block number        41700137
contract address    0xCb7d52d7832969db7b5620aa29bC4bbFd757E67E
from                0xF3d7F77c9eC7b70CaA0ea31EDc0DEBAc38c3a299
to                  BallotBox.(constructor)
gas                 1071142 gas
transaction cost    1071142 gas
input               0x608...00000
decoded input       {
                            "string _electionName": "My Election",
                            "uint256 _registrationDeadline": "1709347200",
                            "uint256  votingDeadline": "1709350800",
```



```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

F:\19_Problem_Statement_19_Ballot\Ballot\ballot\src\Page>npm install
npm WARN deprecated @babel/plugin-proposal-numeric-separator@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintain
ed. Please use @babel/plugin-transform-numeric-separator instead.
npm WARN deprecated @babel/plugin-proposal-nullish-coalescing-operator@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longe
r maintained. Please use @babel/plugin-transform-nullish-coalescing-operator instead.
npm WARN deprecated @babel/plugin-proposal-class-properties@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintaine
d. Please use @babel/plugin-transform-class-properties instead.
npm WARN deprecated @babel/plugin-proposal-private-methods@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained
. Please use @babel/plugin-transform-private-methods instead.
npm WARN deprecated @babel/plugin-proposal-optional-chaining@7.21.0: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintain
ed. Please use @babel/plugin-transform-optional-chaining instead.
npm WARN deprecated stable@0.1.8: Modern JS already guarantees Array#sort() is a stable sort, so this library is deprecated. See the compatibility table on MDN: https:/
/developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#browser_compatibility
npm WARN deprecated rollup-plugin-terser@7.0.2: This package has been deprecated and is no longer maintained. Please use @rollup/plugin-terser
npm WARN deprecated sourcemap-codec@1.4.8: Please use @jridgewell/sourcemap-codec instead
npm WARN deprecated w3c-hr-time@1.0.2: Use your platform's native performance.now() and performance.timeOrigin.
npm WARN deprecated workbox-cacheable-response@6.6.0: workbox-background-sync@6.6.0
npm WARN deprecated svgo@1.3.2: This SVGO version is no longer supported. Upgrade to v2.x.x.
[                  ] \ reify:es-iterator-helpers: timing reifyNode:node_modules/workbox-precaching Completed in 250944ms
```



localhost:3000 says

0x10e15a4c3f97f4611d4e0197d46d6fa385d70a690b24718b6905623131fd4d3f

MetaMask

Confirmed transaction
Transaction 2 confirmed! View on Mumbai Polygonscan
via Microsoft Edge

10.ADVANTAGES&DISADVANTAGES:

10.1 Advantages:

Efficiency and Automation:

Smart contracts automate various financial processes, reducing the need for manual intervention and increasing operational efficiency.

Transparency:

The use of blockchain ensures transparency in transactions, providing a clear and immutable record of all financial activities.

Reduced Fraud:

The tamper-resistant nature of blockchain technology reduces the risk of fraud and unauthorized activities within the financial system.

Faster Transactions:

Smart contracts enable near-instantaneous settlement of transactions, reducing the time and cost associated with traditional banking processes.

Financial Inclusion:

Digital currencies and smart contracts can improve financial inclusion by providing access to banking services for underserved populations.

10.2Disadvantages:

Technical Complexity:

Implementing and maintaining a central bank smart contract system requires a deep understanding of blockchain technology, which may be a barrier for some institutions.

Security Concerns:

Smart contracts are not immune to security vulnerabilities, and any flaws in the code could lead to financial losses or unauthorized access.

Regulatory Uncertainty:

The regulatory landscape for blockchain and digital currencies is still evolving, and uncertainties may pose challenges for central banks in terms of compliance.

Scalability Issues:

As transaction volumes increase, scalability can become a concern, requiring careful design to handle a growing number of transactions.

11.CONCLUSION:

Central bank smart contracts offer a promising avenue for enhancing the efficiency, transparency, and security of monetary transactions. By leveraging blockchain technology, these contracts can automate various financial processes, reducing the risk of errors and fraud. Additionally, the decentralized nature of smart contracts ensures a transparent and tamper-resistant system.

However, challenges such as scalability, regulatory concerns, and the need for a robust security framework must be addressed to fully realize the potential benefits. Striking the right balance between innovation and risk management is crucial for the successful implementation of central bank smart contracts.

12.FUTURESCOPE:

Decentralized Monetary Policy Execution: Smart contracts could automate the execution of monetary policies, adjusting interest rates or money supply based on predefined conditions. This could lead to more efficient and transparent monetary policy implementation.

Real-Time Economic Data Analysis: Smart contracts could be programmed to analyze real-time economic data. This would enable central banks to make quicker and more informed decisions based on the most current information available.

Cross-Border Transaction: Central bank digital currencies (CBDCs) integrated with smart contracts could streamline cross-border transactions. Smart contracts could automatically enforce compliance with international regulations, reducing the need for intermediaries and increasing the speed of transactions.

1. APPENDIX

13.1SourceCode:

```
// 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 mint Money(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];
    }
}


}
```

13.2 GitHub&ProjectDemoLink: