

Project Development Phase and Model Performance Test

Date	03 November 2023
Team ID	NM2023TMID11290
Project Name	Central Bank Smart Contract

Model Performance Testing:

Project team shall fill the following information when working for blockchain.

S.No.	Parameter	Values	Screenshot
1.	Information gathering	Setup all the Prerequisite:	The Screenshots of information gathering are all given below.
2.	Extract the zip files	Open to vs code	The Screenshots of extract the zip file are all given below.

3.	Remix Idle Platform explorting	Deploy the smart contract code Deploy and run the transaction. Byselecting the environment - inject the MetaMask.	The Screenshots of extract the zip file are all given below.
4	Open file explorer	Open the extracted file and click onthe folder. Open src, and search for utiles. Open cmd enter commands 1.npm install 2.npm bootstrap 3.npm start	The Screenshots of extract the zip file are all given below.
5	LOCALHOST IPADDRESS	Copy the address and open it to chrome so you can see the front end of your project.	The Screenshots of extract the zip file are all given below.

1. INFORMATION GATHERING

Screenshot 1

➔

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you're focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) ➔

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

PROBLEM

The regulatory landscape for CBSCs is still evolving, and there is a lack of clarity on how they will be regulated. This uncertainty could make it difficult for central banks to implement CBSCs and could also discourage private sector participation in the CBSC ecosystem.

Key rules of brainstorming

To run a smooth and productive session

🗨️ Stay in topic.

💡 Encourage wild ideas.

⏸️ Defer judgment.

👂 Listen to others.

🗨️ Go for volume.

👁️ If possible, be visual.

Screenshot 2

2

Brainstorm

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

🕒 10 minutes

TP

You can select a sticky note and hit the search (magnifying glass) icon to start grouping!

Bestir Reserve

Avvin Barito

Banivizhal

Biroth

2

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 can break it up into smaller sub-groups.

🕒 20 minutes

TP

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

CBSCs can automate monetary policy operations, such as open market operations, interest rate setting, and reserve requirements.

CBSCs can be used to promote financial stability by implementing macroprudential policies.

CBSCs can be used to oversee and regulate the financial system

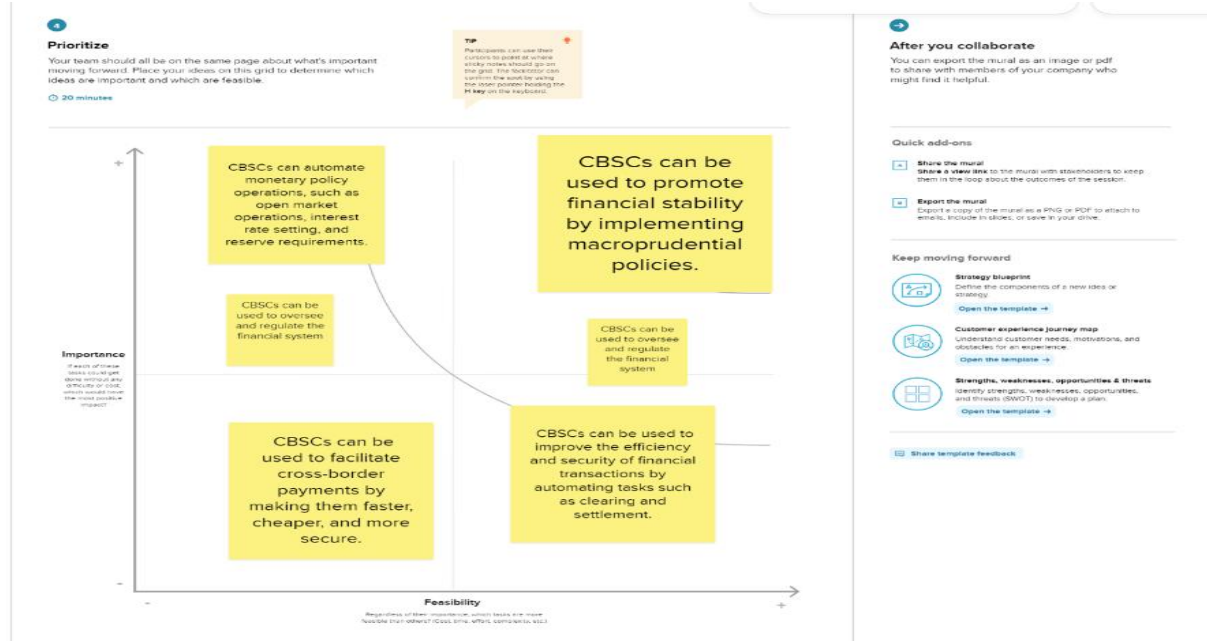
CBSCs can issue and manage CBDCs

CBSCs can be used to collect data

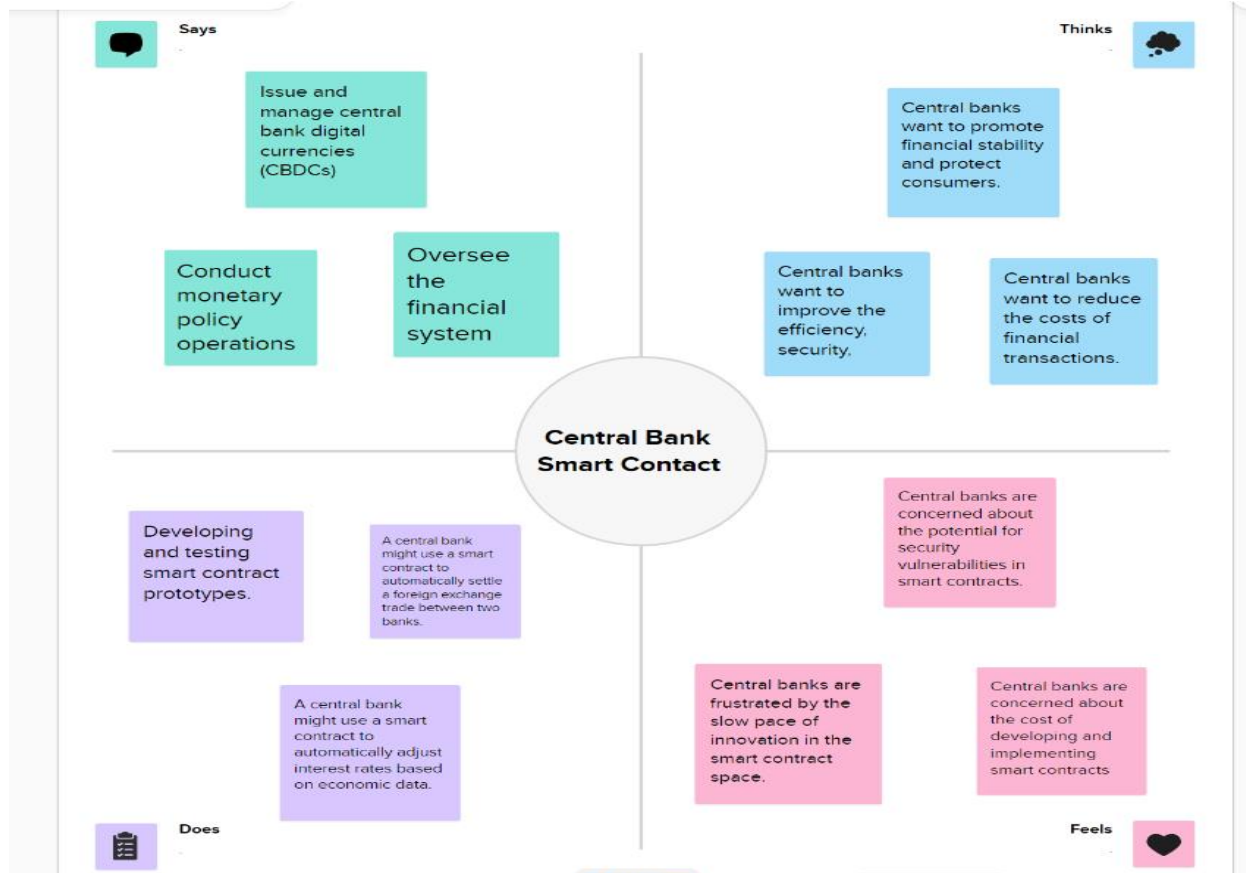
CBSCs can be used to facilitate cross-border payments by making them faster, cheaper, and more secure.

CBSCs can be used to improve the efficiency and security of financial transactions by automating tasks such as clearing and settlement.

Screenshot 3

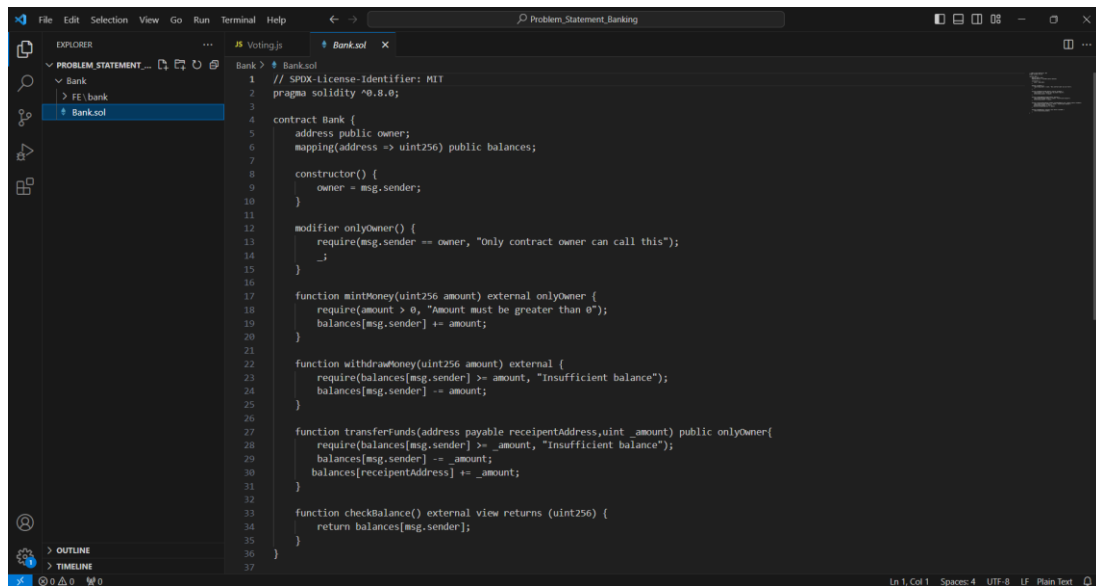


Screenshot 4



2. EXTRACT THE ZIP FILE

Screenshot 1

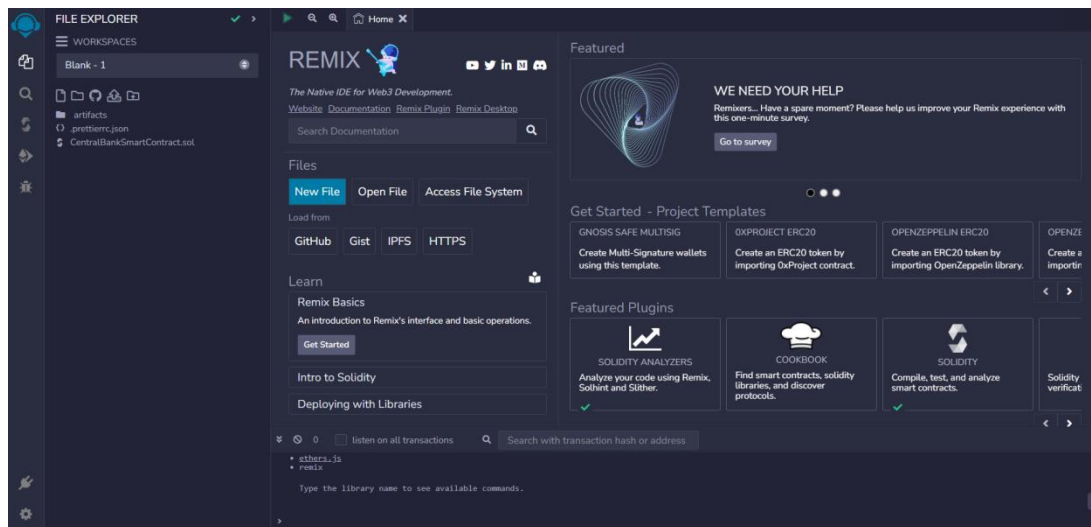


The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a folder named 'Bank' containing a file 'Bank.sol'. The code editor displays the content of 'Bank.sol', which is a Solidity contract named 'Bank'. The contract includes a constructor, a modifier 'onlyOwner', and three functions: 'mintMoney', 'withdrawMoney', and 'transferFunds'. The 'mintMoney' function is marked as 'external' and 'onlyOwner'. The 'withdrawMoney' function is marked as 'external'. The 'transferFunds' function is marked as 'public' and 'onlyOwner'. The 'checkBalance' function is marked as 'external' and 'view'.

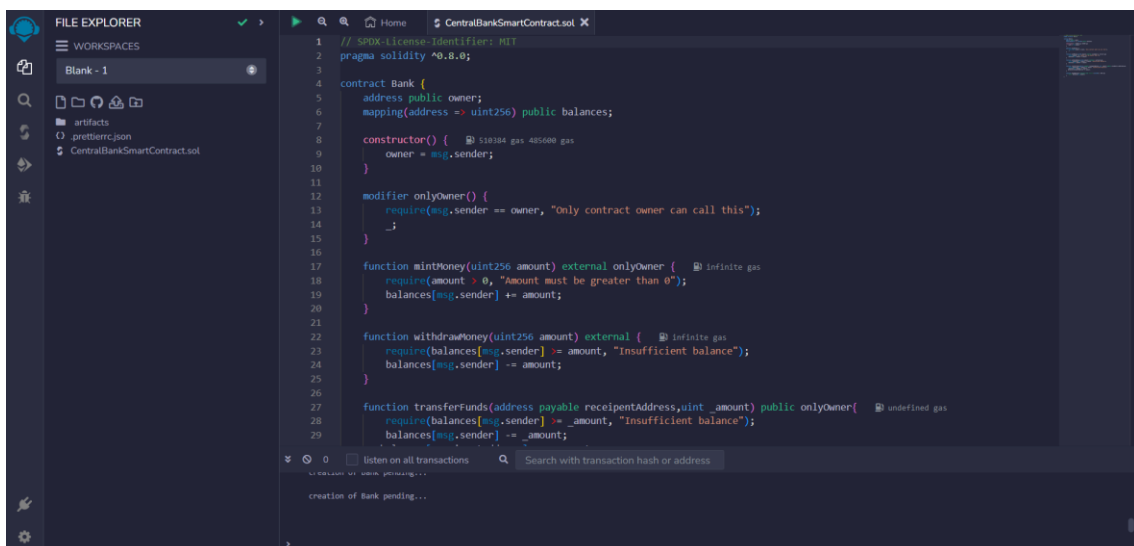
```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract Bank {
5     address public owner;
6     mapping(address => uint256) public balances;
7
8     constructor() {
9         owner = msg.sender;
10    }
11
12    modifier onlyOwner() {
13        require(msg.sender == owner, "Only contract owner can call this");
14        _;
15    }
16
17    function mintMoney(uint256 amount) external onlyOwner {
18        require(amount > 0, "Amount must be greater than 0");
19        balances[msg.sender] += amount;
20    }
21
22    function withdrawMoney(uint256 amount) external {
23        require(balances[msg.sender] >= amount, "Insufficient balance");
24        balances[msg.sender] -= amount;
25    }
26
27    function transferFunds(address payable recipientAddress, uint _amount) public onlyOwner {
28        require(balances[msg.sender] >= _amount, "Insufficient balance");
29        balances[msg.sender] -= _amount;
30        balances[recipientAddress] += _amount;
31    }
32
33    function checkBalance() external view returns (uint256) {
34        return balances[msg.sender];
35    }
36
37 }
```

3. Remix Ide platform Explorting

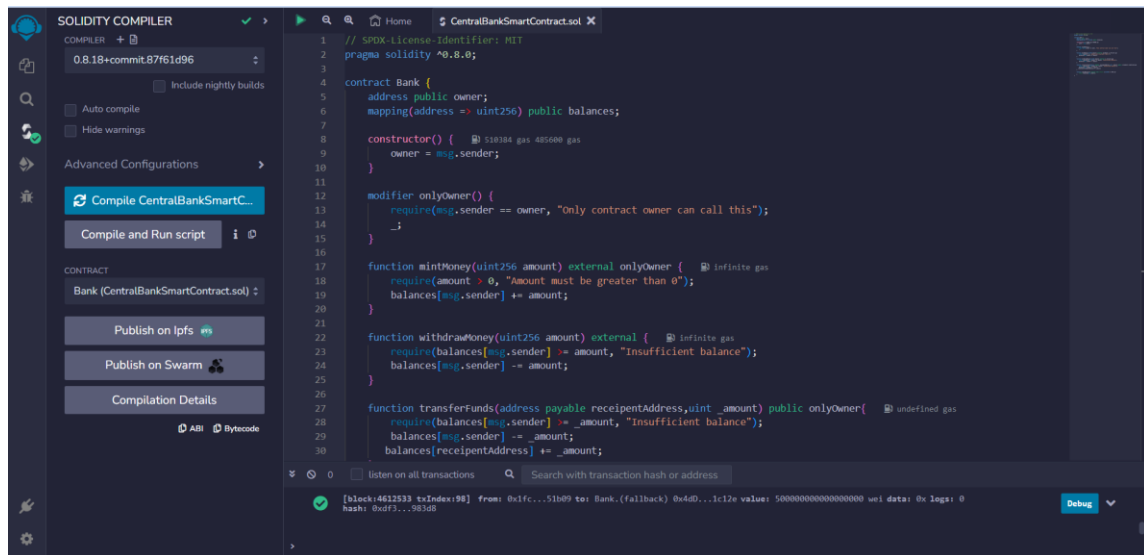
Screenshot 1



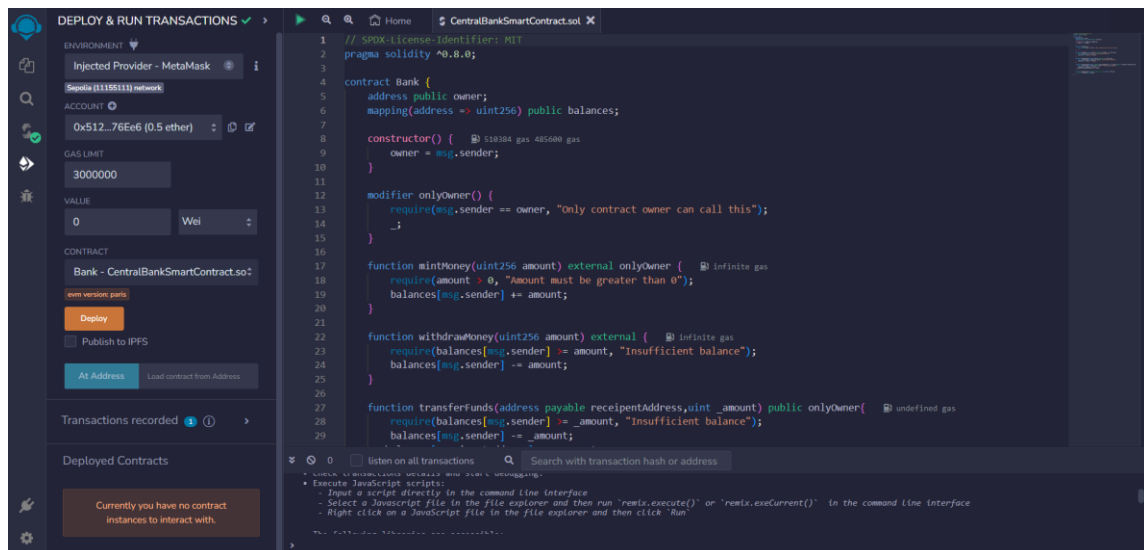
Screenshot 2



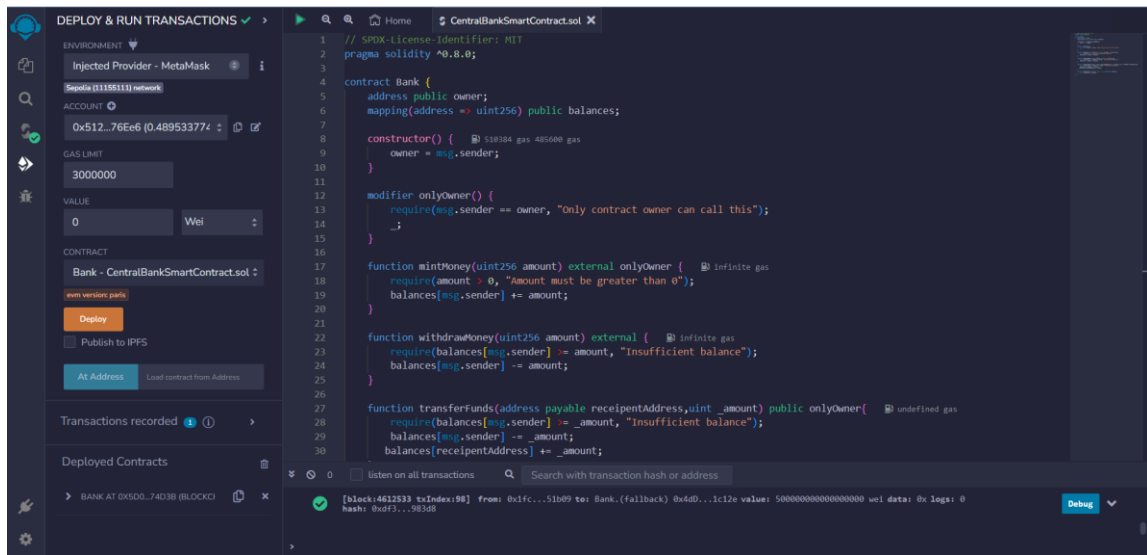
Screenshot 3



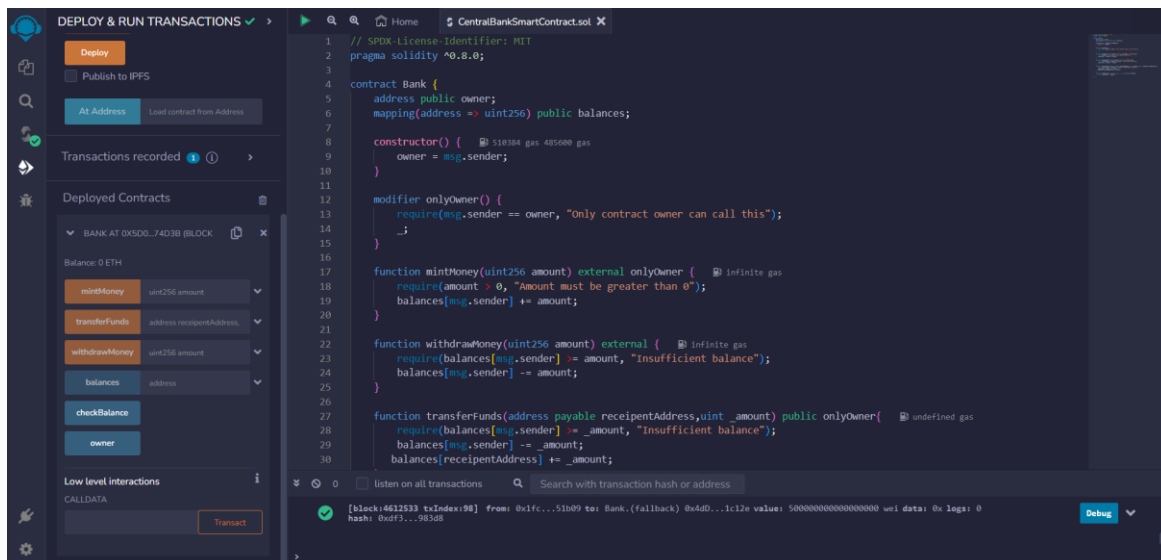
Screenshot 4



Screenshot 5

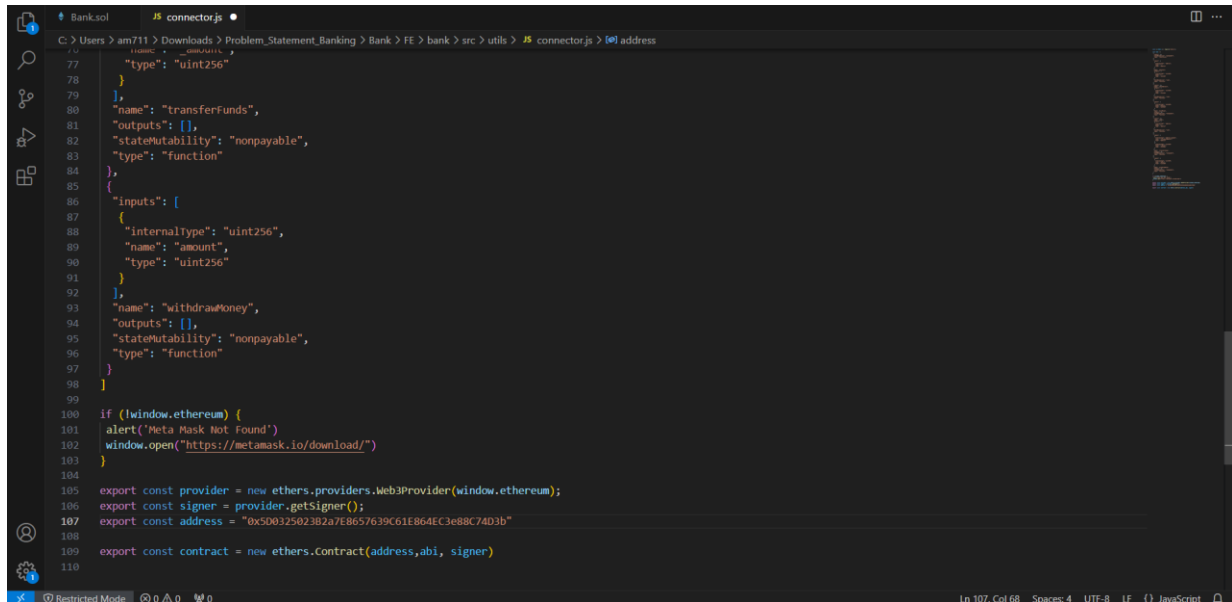


Screenshot 6



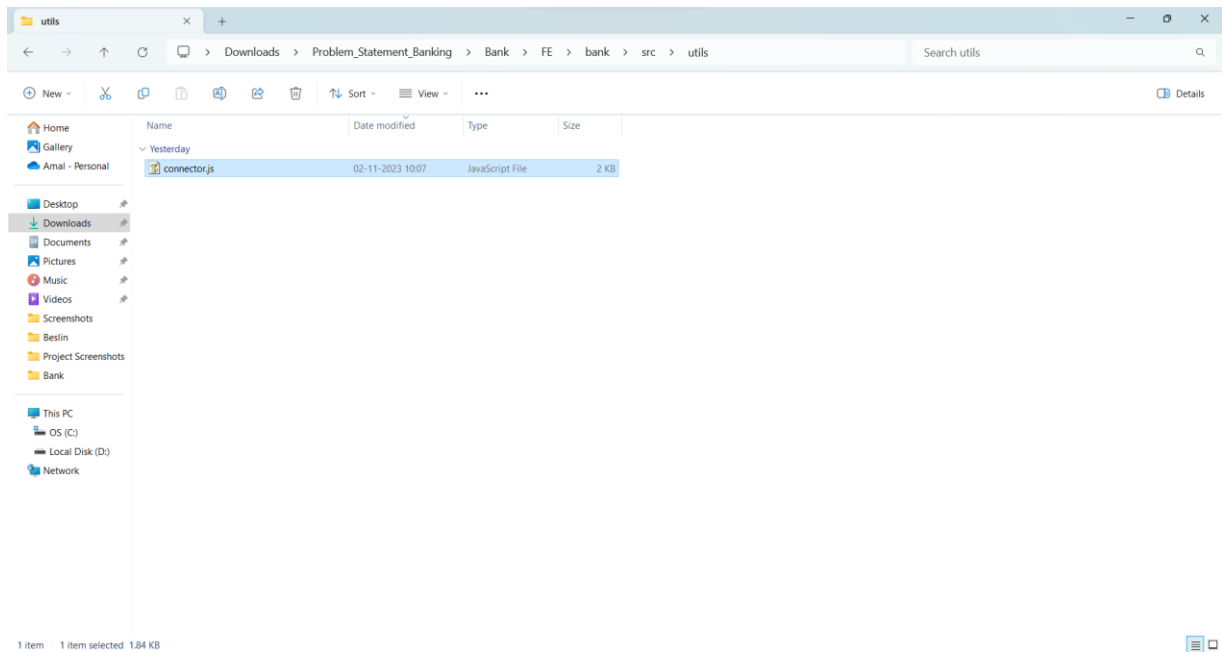
4. OPEN THE FILE EXPLORER

Screenshot 1

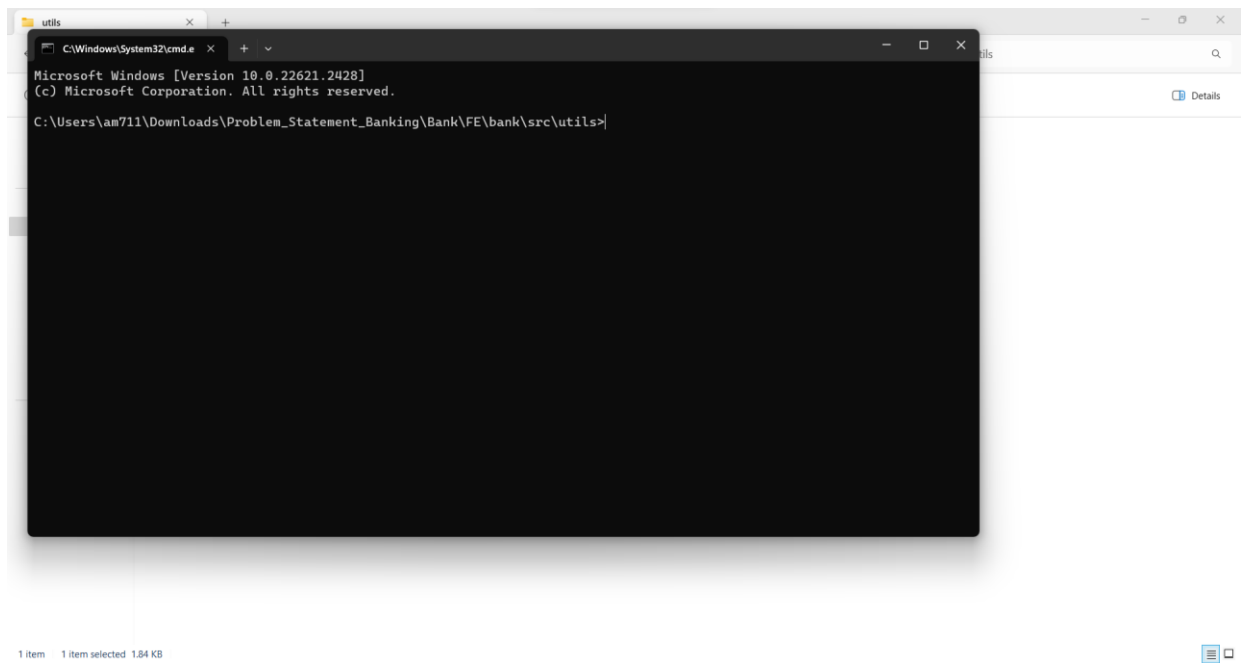


```
77     "type": "uint256"
78   },
79   ],
80   "name": "transferFunds",
81   "outputs": [],
82   "stateMutability": "nonpayable",
83   "type": "function"
84 },
85 {
86   "inputs": [
87     {
88       "internalType": "uint256",
89       "name": "amount",
90       "type": "uint256"
91     }
92   ],
93   "name": "withdrawMoney",
94   "outputs": [],
95   "stateMutability": "nonpayable",
96   "type": "function"
97 }
98 ]
99
100 if (!window.ethereum) {
101   alert("Meta Mask Not Found")
102   window.open("https://metamask.io/download/")
103 }
104
105 export const provider = new ethers.providers.Web3Provider(window.ethereum);
106 export const signer = provider.getSigner();
107 export const address = "0x5D0325023B2a7E8657639C61E864EC3e88C74D3b";
108
109 export const contract = new ethers.Contract(address, abi, signer);
110
```

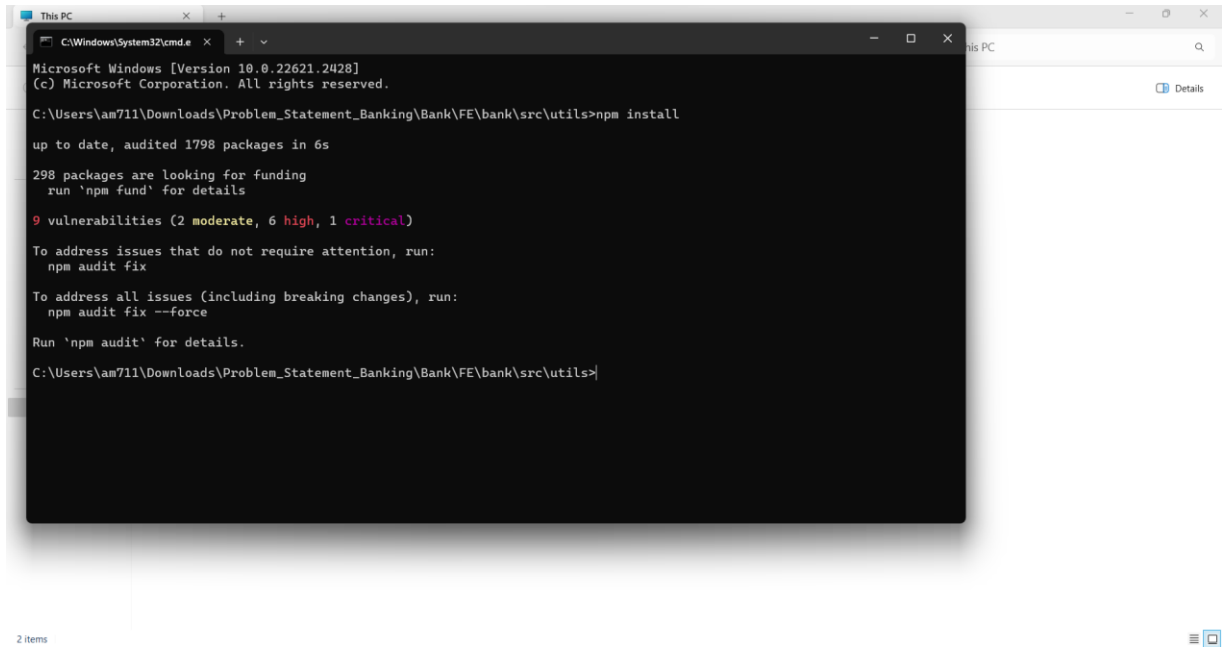
Screenshot 2



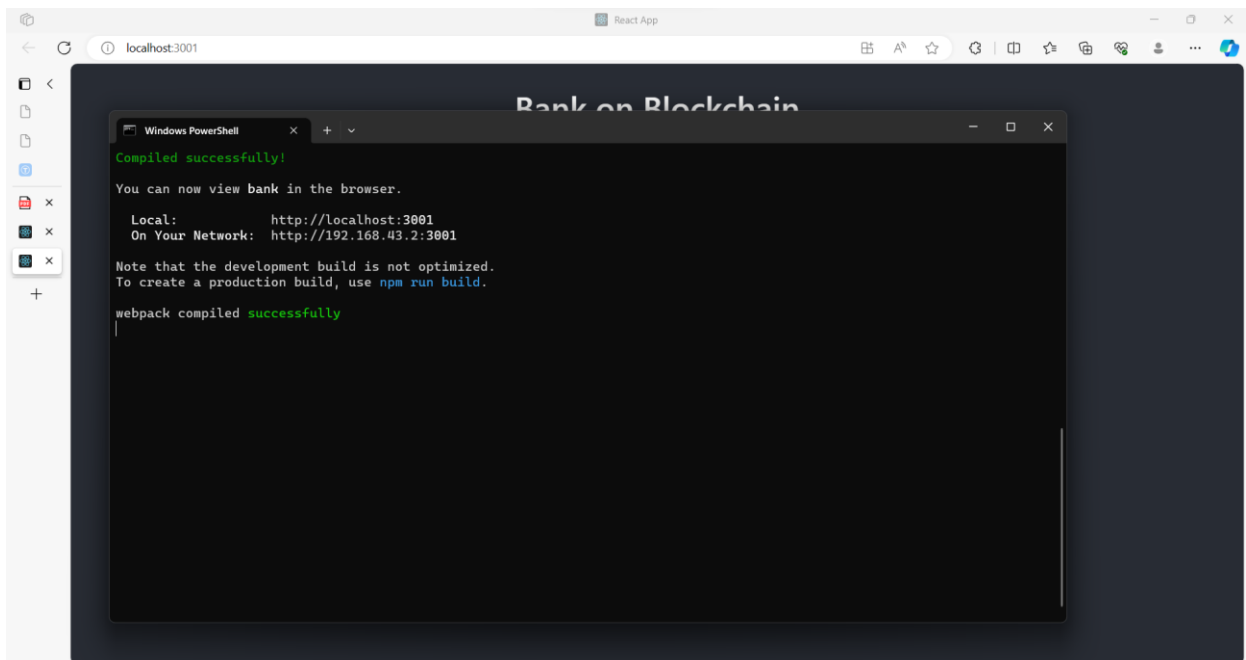
Screenshot 3



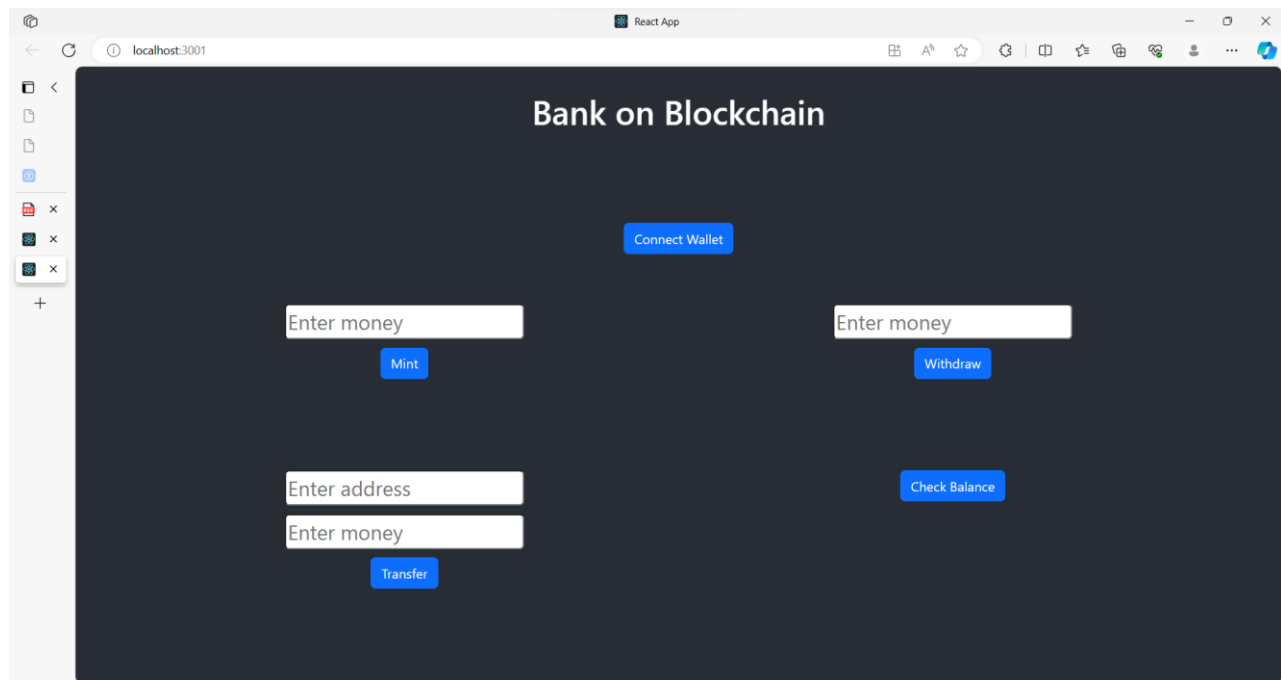
Screenshot 4



Screenshot 5



5. LOCALHOST IPADDRESS



<http://localhost:3001>