A Simple Bank smart contract code in Solidity programming language

Introduction:

Blockchain technology has revolutionized various industries, including banking, by introducing decentralized and secure solutions. One of the fundamental applications of blockchain in banking is the development of smart contracts to automate financial processes. This report aims to explore the development of a Simple Bank smart contract using the Solidity programming language.

Overview of Blockchain Technology in Banking

Blockchain technology, a distributed ledger technology, enables secure and transparent transactions without the need for intermediaries. It has found significant applications in various sectors, including banking. In banking, blockchain enhances the efficiency, security, and transparency of financial transactions.

Need for Smart Contracts in Banking

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They automate financial processes, reduce the need for intermediaries, and enhance transparency and security. In banking, smart contracts can automate tasks such as loan processing, fund transfers, and more.

Understanding Simple Bank Smart Contracts

A Simple Bank smart contract is a contract designed to handle basic banking functions such as deposits, withdrawals, loans, repayments, and peer-to-peer transactions.

Key Features:

- **Deposit:** Users can deposit funds into their accounts.

```
function deposit() public payable {

balances[msg.sender] += msg.value;

emit Deposit(msg.sender, msg.value);

}
```

- **Withdrawal:** Users can withdraw funds from their accounts.

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

- **Loan:** Users can borrow loans from the bank.

```
function borrow(uint256 amount) public {
    require(balances[msg.sender] >= amount / 2, "Collateral required");
    loans[msg.sender] += amount;
    balances[msg.sender] += amount;
    emit Borrow(msg.sender, amount);
  }
```

- **Repayment:** Users can repay their loans.

```
function repay(uint256 amount) public {
    require(loans[msg.sender] >= amount, "Loan amount exceeded");
    require(balances[msg.sender] >= amount, "Insufficient balance");
    loans[msg.sender] -= amount;
    balances[msg.sender] -= amount;
    emit Repay(msg.sender, amount);
  }
```

- **Peer-to-Peer Transactions:** Users can transfer funds to other users.

```
function transfer(address to, uint256 amount) public {
    require(balances[msg.sender] >= amount, "Insufficient balance");
    balances[msg.sender] -= amount;
    balances[to] += amount;
    emit Transfer(msg.sender, to, amount);
  }
```

- **Check Balance:** Users can check their balance.

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

- **Check Loan Balance:** Users can check how much loan should they repay.

```
function getLoanBalance() public view returns (uint256) {
      return loans[msg.sender];
  }
```

Developing a Simple Bank Smart Contract

Solidity Programming Language:

Solidity is a statically-typed programming language designed for developing smart contracts on the Ethereum blockchain.

Code Development:

All the fuctions that was used to develop the smart contract for the bank has be mentioned in the previous section

Tools Used:

- **Truffle**: A development framework for Ethereum that provides tools for compiling, deploying, and testing smart contracts. It was used for compiling and migrating the Simple Bank smart contract.
- **Node.js and npm**: JavaScript runtime and package manager, respectively, used for managing dependencies and running scripts in the project.
- **Ganache**: A personal blockchain for Ethereum development, used for testing and deploying smart contracts locally.
- **VS Code**: An integrated development environment (IDE) commonly used for writing, editing, and debugging code. It was likely used for writing and editing the Solidity smart contract code, as well as for managing the project files.

Deployment of the Smart Contract

Deployment:

- Deploy the smart contract using tools like Remix or Truffle.
- Example deployment to Ganache (a local blockchain):

  ```shell
  truffle migrate --network development
  ```

Interacting with the Deployed Smart Contract

Interacting via Web3.js:

```
import Web3 from 'web3';

import SimpleBankABI from './SimpleBankABI.json';

const web3 = new Web3(Web3.givenProvider || process.env.REACT_APP_GANACHE_URL);

const contractAddress = process.env.REACT_APP_CONTRACT_ADDRESS;

const simpleBankContract = new web3.eth.Contract(SimpleBankABI.abi, contractAddress);

export { web3, simpleBankContract };
```

Application of Simple Bank Smart Contracts

Overview of Simple Bank Smart Contracts:

A Simple Bank smart contract is designed to handle basic banking functions such as deposits, withdrawals, loans, repayments, and peer-to-peer transactions. These contracts are self-executing, with the terms of the agreement directly written into the code, and they operate on the Ethereum blockchain.

Practical Applications in the Banking Sector:

Deposits and Withdrawals: Users can deposit and withdraw funds securely without needing a physical bank.

Loans and Repayments: Smart contracts can automate the loan process, including the disbursement and repayment of loans, reducing the need for intermediaries.

Peer-to-Peer Transactions: Users can transfer funds directly to other users, enabling efficient and low-cost money transfers.

Savings and Investment: Smart contracts can be used to create decentralized savings and investment accounts, offering interest based on predefined conditions.

Remittances: Facilitates cross-border remittances, making them faster and cheaper than traditional methods.

Advantages of Using Simple Bank Smart Contracts

Automation and Efficiency:

Smart contracts automate financial processes, eliminating the need for manual intervention. This reduces processing times and minimizes the risk of human error. For example, loan approvals and repayments can be automated based on predefined conditions.

Transparency and Trust:

The code and transactions are visible on the blockchain, ensuring transparency. This builds trust among users as they can verify the terms and conditions of the contract and track all transactions.

Security and Immutability:

Smart contracts are secure and tamper-proof due to the underlying blockchain technology. Once deployed, the contract code cannot be altered, ensuring the integrity of the agreement.

Cost Reduction:

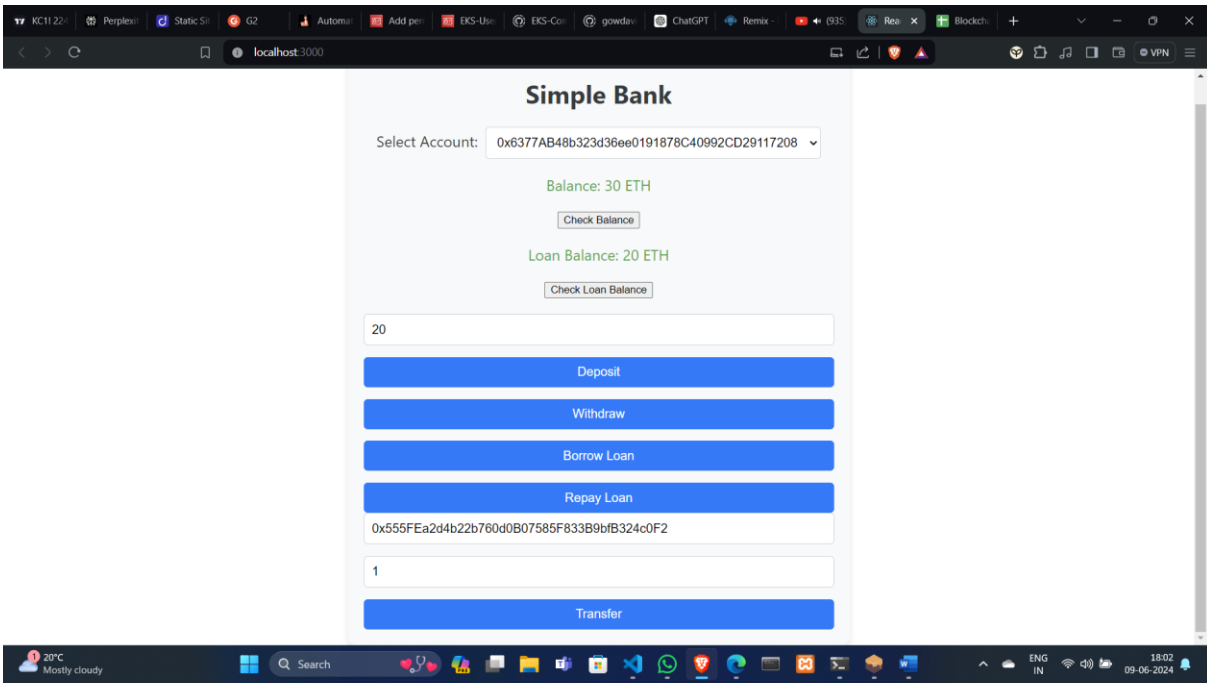
By eliminating intermediaries and automating processes, smart contracts reduce operational costs. This is particularly beneficial for financial institutions and customers who can save on fees and commissions.

Accessibility and Inclusion:

Smart contracts provide banking services to unbanked and underbanked populations by enabling access to financial services through internet-enabled devices. This promotes financial inclusion and empowers individuals in remote areas.

Result Screenshot

React Page –



Simple Bank

Select Account: 0x6377AB48b323d36ee0191878C40992CD29117208

Balance: 30 ETH

Check Balance

Loan Balance: 20 ETH

Check Loan Balance

20

Deposit

Withdraw

Borrow Loan

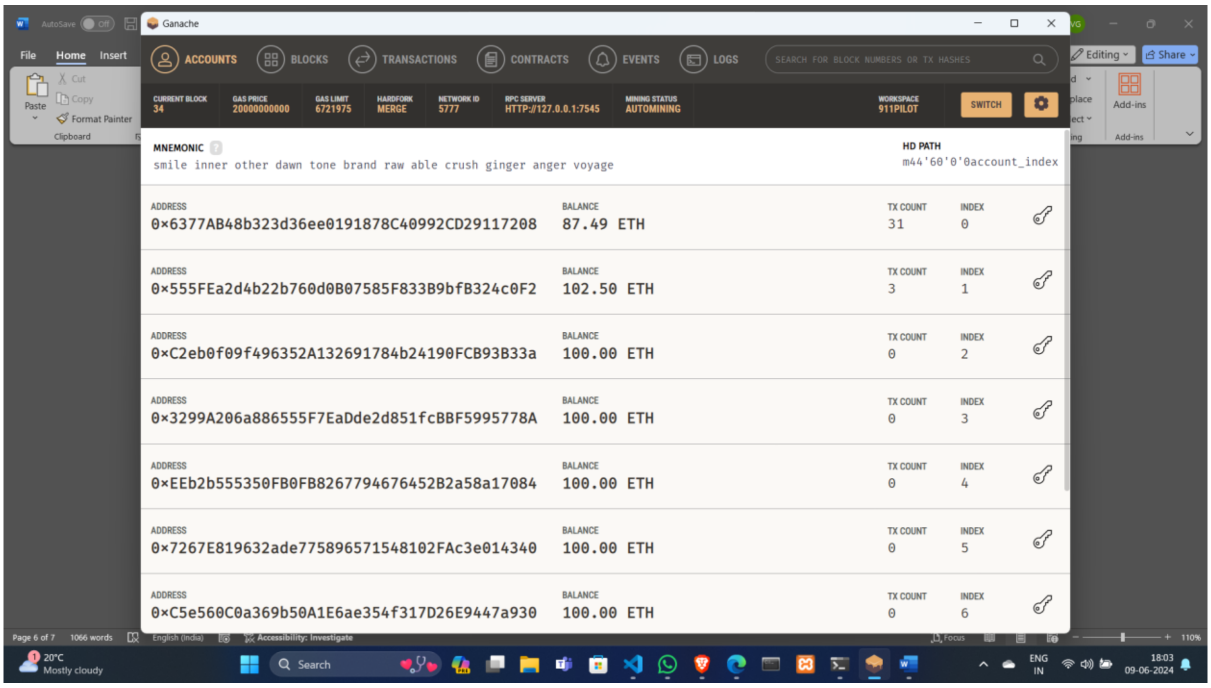
Repay Loan

0x555FEa2d4b22b760d0B07585F833B9bfB324c0F2

1

Transfer

Ganache –



```
ACCOUNTS   BLOCKS   TRANSACTIONS   CONTRACTS   EVENTS   LOGS

CURRENT BLOCK  GAS PRICE    GAS LIMIT  HARDFORK  NETWORK ID  RPC SERVER              MINING STATUS  WORKSPACE
34             20000000000  6721975    MERGE     5777        HTTP://127.0.0.1:7545   AUTOMINING     911PILOT

MNEMONIC                                                           HD PATH
smile inner other dawn tone brand raw able crush ginger anger voyage   m44'60'0'0account_index

ADDRESS                                      BALANCE      TX COUNT  INDEX
0×6377AB48b323d36ee0191878C40992CD29117208   87.49 ETH    31        0
0×555FEa2d4b22b760d0B07585F833B9bfB324c0F2   102.50 ETH   3         1
0×C2eb0f09f496352A132691784b24190FCB93B33a   100.00 ETH   0         2
0×3299A206a886555F7EaDde2d851fcBBF5995778A   100.00 ETH   0         3
0×EEb2b555350FB0FB8267794676452B2a58a17084   100.00 ETH   0         4
0×7267E819632ade775896571548102FAc3e014340   100.00 ETH   0         5
0×C5e560C0a369b50A1E6ae354f317D26E9447a930   100.00 ETH   0         6
```

Conclusion

The integration of blockchain technology and smart contracts within the banking sector signifies a transformative step towards creating efficient, transparent, and secure financial systems. The Simple Bank smart contract, developed using Solidity, automates fundamental banking services like deposits, withdrawals, loans, and peer-to-peer transactions, enhancing operational efficiency by eliminating intermediaries and reducing processing times.

Blockchain's transparency fosters trust, as all transactions and contract terms are verifiable by participants. The immutability of smart contracts ensures security by preventing unauthorized modifications, while cost reductions stem from streamlined operations and lower fees.

Moreover, smart contracts promote financial inclusion by providing access to banking services for unbanked and underbanked populations via internet-enabled devices. This inclusivity empowers individuals in remote areas to participate in the global financial system.

In conclusion, the Simple Bank smart contract exemplifies the potential of blockchain to revolutionize banking, offering significant benefits in efficiency, security, and inclusivity.