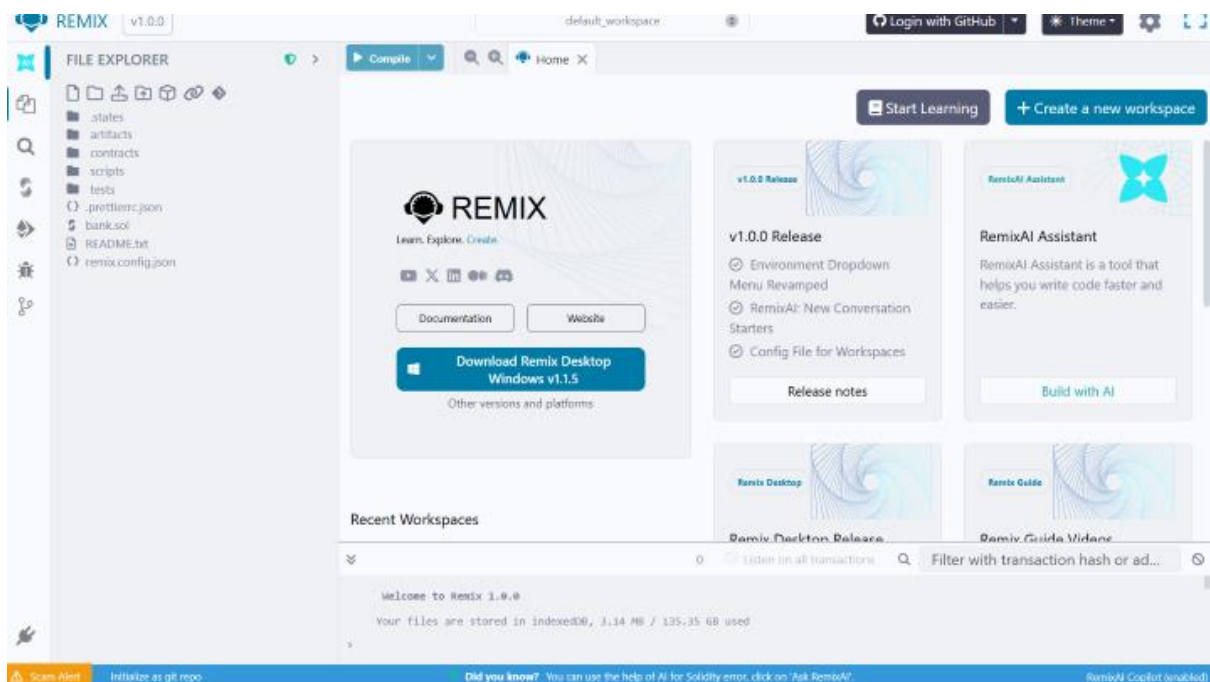


ASSIGNMENT NO:03

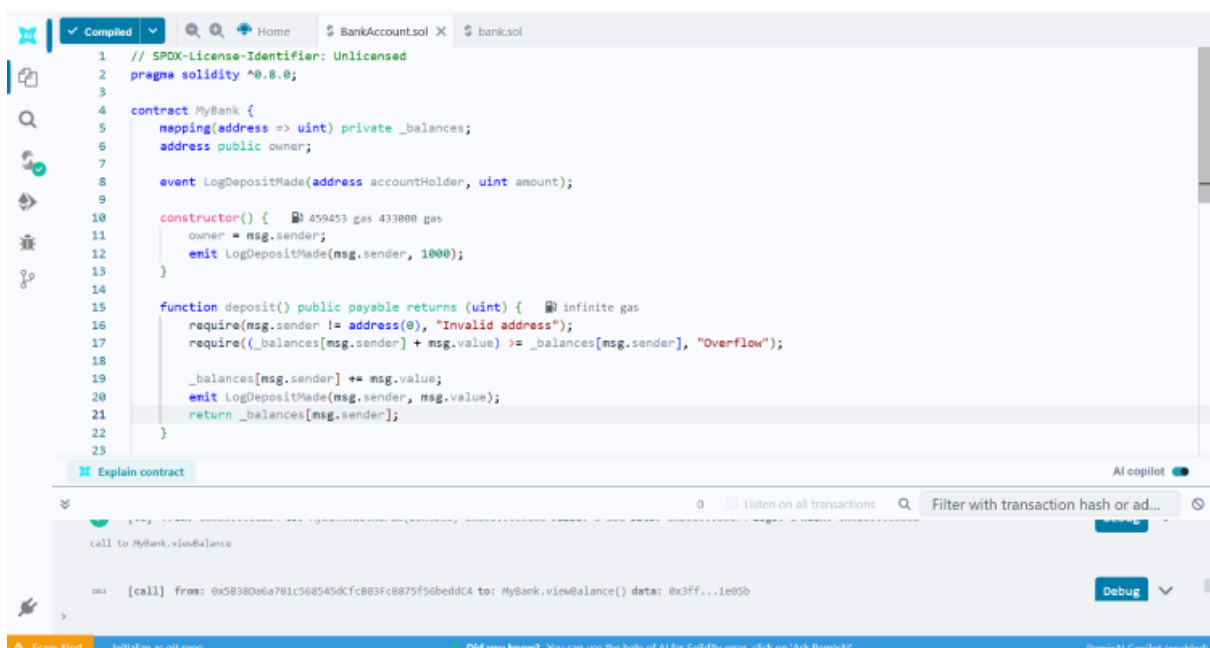
AIM: Write a smart contract on a test network, for Bank account of a customer for following operations:

- Deposit money
- Withdraw Money
- Show balance

STEP 1:



STEP 2:



STEP 3:

The screenshot shows the Remix IDE with the `bank.sol` file open. The code defines a `HyBank` contract with a `_balances` mapping, an `owner`, and a `LogDepositMade` event. The `constructor` initializes the owner and emits a log for a 1000 unit deposit. The `deposit` function checks for a valid address and sufficient balance, then updates the balance and emits a log. The AI Copilot panel at the bottom shows a call to `HyBank.viewBalance` with a debug button.

```
1 // SPDX-License-Identifier: Unlicensed
2 pragma solidity ^0.8.0;
3
4 contract HyBank {
5     mapping(address => uint) private _balances;
6     address public owner;
7
8     event LogDepositMade(address accountHolder, uint amount);
9
10    constructor() {
11        owner = msg.sender;
12        emit LogDepositMade(msg.sender, 1000);
13    }
14
15    function deposit() public payable returns (uint) {
16        require(msg.sender != address(0), "Invalid address");
17        require(_balances[msg.sender] + msg.value >= _balances[msg.sender], "Overflow");
18
19        _balances[msg.sender] += msg.value;
20        emit LogDepositMade(msg.sender, msg.value);
21        return _balances[msg.sender];
22    }
23 }
```

STEP 4:

The screenshot shows the Remix IDE with the `bank.sol` file open. The code adds a `withdraw` function that checks for a valid address and sufficient balance, then transfers the amount and emits a log. It also adds a `viewBalance` function that returns the current balance. The AI Copilot panel at the bottom shows a call to `HyBank.viewBalance` with a debug button.

```
23
24 function withdraw(uint withdrawAmount) public returns (uint) {
25     require(msg.sender != address(0), "Invalid address");
26     require(_balances[msg.sender] >= withdrawAmount, "Insufficient balance");
27
28     _balances[msg.sender] -= withdrawAmount;
29     payable(msg.sender).transfer(withdrawAmount);
30     emit LogDepositMade(msg.sender, withdrawAmount); // You might want to use a different event here
31     return _balances[msg.sender];
32 }
33
34 function viewBalance() public view returns (uint) {
35     return _balances[msg.sender];
36 }
37
38 }
```

STEP 5:

- Deploy:

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active. It shows the account '0x583...eddC4 (99.999999999999999999)' and the gas limit '3000000'. The contract 'MyBank - BankAccount.sol' is selected. The 'Deploy' button is highlighted. Below it, the 'At Address' button is visible. The main editor shows the Solidity code for 'MyBank'.

```
1 // SPDX-License-Identifier: Unlicensed
2 pragma solidity ^0.8.0;
3
4 contract MyBank {
5     mapping(address => uint) private _balances;
6     address public owner;
7
8     event LogDepositMade(address accountHolder, uint amount);
9
10    constructor() {
11        459453 gas 433000 gas
12        owner = msg.sender;
13        emit LogDepositMade(msg.sender, 1000);
14    }
15
16    function deposit() public payable returns (uint) {
17        require(msg.sender != address(0), "Invalid address");
18        require((_balances[msg.sender] + msg.value) >= _balances[msg.sender], "Overflow");
19        _balances[msg.sender] += msg.value;
20    }
21}
```

The console shows the deployment transaction: '[vm] from: 0x583...eddC4 to: MyBank.(constructor) value: 0 wei data: 0x088...e0033 logs: 1 hash: 0x077...53a92'. The status is 'creation of MyBank pending...'. The bottom status bar indicates 'RemixAI Copilot (enabled)'.

- Deposit money:

The screenshot shows the Remix IDE interface after deployment. The 'DEPLOY & RUN TRANSACTIONS' panel is active. It shows the account '0x583...eddC4 (99.999999999999999999)' and the gas limit '3000000'. The contract 'MyBank - BankAccount.sol' is selected. The 'deposit' button is highlighted. Below it, the 'At Address' button is visible. The main editor shows the Solidity code for 'MyBank'.

```
2 pragma solidity ^0.8.0;
3
4 contract MyBank {
5     mapping(address => uint) private _balances;
6     address public owner;
7
8     event LogDepositMade(address accountHolder, uint amount);
9
10    constructor() {
11        459453 gas 433000 gas
12        owner = msg.sender;
13        emit LogDepositMade(msg.sender, 1000);
14    }
15
16    function deposit() public payable returns (uint) {
17        require(msg.sender != address(0), "Invalid address");
18        require((_balances[msg.sender] + msg.value) >= _balances[msg.sender], "Overflow");
19        _balances[msg.sender] += msg.value;
20    }
21}
```

The console shows the deposit transaction: '[vm] from: 0x583...eddC4 to: MyBank.deposit() 0x091...39138 value: 1000 wei data: 0x00e...30db8 logs: 1 hash: 0x0cd...24309'. The status is 'transaction to MyBank.deposit pending...'. The bottom status bar indicates 'RemixAI Copilot (enabled)'.

- **Show balance after Deposit money:**

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible. Under 'Deployed Contracts', the 'MYBANK' contract is listed with a balance of 0.000000000000000000 ETH. The 'deposit' button is highlighted. The main editor displays the Solidity code for the 'MyBank' contract. The bottom panel shows the transaction log with a successful deposit transaction.

```

2  pragma solidity ^0.8.0;
3
4  contract MyBank {
5      mapping(address => uint) private _balances;
6      address public owner;
7
8      event LogDepositMade(address accountHolder, uint amount);
9
10     constructor() {
11         owner = msg.sender;
12         emit LogDepositMade(msg.sender, 1000);
13     }
14
15     function deposit() public payable returns (uint) {
16         require(msg.sender != address(0), "Invalid address");
17         require((_balances[msg.sender] + msg.value) >= _balances[msg.sender], "Overflow");
18     }

```

The transaction log shows a successful deposit transaction with the following details:

- hash: 0x480...2866f
- transaction to MyBank.deposit pending ...
- [vm] from: 0x583...eddC4 to: MyBank.deposit() @0x91...39138 value: 1000 wei data: 0x0e...30b0 logs: 1
- hash: 0xccc...24309
- call to MyBank.viewBalance
- [call] from: 0x58380a6a701c568545dcfcb03fc8b75f56beddC4 to: MyBank.viewBalance() data: 0x3ff...1e05b

- **Withdraw Money**

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible. Under 'Deployed Contracts', the 'MYBANK' contract is listed with a balance of 0.000000000000000000 ETH. The 'withdraw' button is highlighted. The main editor displays the Solidity code for the 'MyBank' contract. The bottom panel shows the transaction log with a successful withdraw transaction.

```

2  pragma solidity ^0.8.0;
3
4  contract MyBank {
5      mapping(address => uint) private _balances;
6      address public owner;
7
8      event LogDepositMade(address accountHolder, uint amount);
9
10     constructor() {
11         owner = msg.sender;
12         emit LogDepositMade(msg.sender, 1000);
13     }
14
15     function deposit() public payable returns (uint) {
16         require(msg.sender != address(0), "Invalid address");
17         require((_balances[msg.sender] + msg.value) >= _balances[msg.sender], "Overflow");
18     }

```

The transaction log shows a successful withdraw transaction with the following details:

- hash: 0xccc...24309
- call to MyBank.viewBalance
- [call] from: 0x58380a6a701c568545dcfcb03fc8b75f56beddC4 to: MyBank.viewBalance() data: 0x3ff...1e05b
- transaction to MyBank.withdraw pending ...
- [vm] from: 0x583...eddC4 to: MyBank.withdraw(uint256) @0x91...39138 value: 0 wei data: 0x2e1...001f4 logs: 1
- hash: 0x3a0...90e38

- **Show Balance after Withdraw Money**

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active, showing a 'withdraw' button with a value of 500. The main editor displays the Solidity code for the 'MyBank' contract. The bottom panel shows the transaction history, including a call to 'MyBank.viewBalance()'.

```

pragma solidity ^0.8.0;

contract MyBank {
    mapping(address => uint) private _balances;
    address public owner;

    event LogDepositMade(address accountHolder, uint amount);

    constructor() {
        459453 gas 433888 gas
        owner = msg.sender;
        emit LogDepositMade(msg.sender, 1000);
    }

    function deposit() public payable returns (uint) {
        require(msg.sender != address(0), "Invalid address");
        require((_balances[msg.sender] + msg.value) >= _balances[msg.sender], "Overflow");
    }
}

```

Transaction history:

- hash: 0x3a8...95e38
- call to MyBank.viewBalance()
- [call] from: 0x58380a6a701c568545dcf803fc8875f56beddC4 to: MyBank.viewBalance() data: 0x3ff...1e05b
- call to MyBank.owner
- [call] from: 0x58380a6a701c568545dcf803fc8875f56beddC4 to: MyBank.owner() data: 0x8da...5cb5b

- **Check Owner:**

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active, showing an 'owner' button. The main editor displays the Solidity code for the 'MyBank' contract. The bottom panel shows the transaction history, including a call to 'MyBank.owner()'.

```

pragma solidity ^0.8.0;

contract MyBank {
    mapping(address => uint) private _balances;
    address public owner;

    event LogDepositMade(address accountHolder, uint amount);

    constructor() {
        459453 gas 433888 gas
        owner = msg.sender;
        emit LogDepositMade(msg.sender, 1000);
    }

    function deposit() public payable returns (uint) {
        require(msg.sender != address(0), "Invalid address");
        require((_balances[msg.sender] + msg.value) >= _balances[msg.sender], "Overflow");
    }
}

```

Transaction history:

- transact to MyBank.withdraw pending ...
- [ve] from: 0x583...eddC4 to: MyBank.withdraw(uint256) 0xd91...39138 value: 0 wei data: 0x2e1...001f4 logs: 1
- hash: 0x3a8...95e38
- call to MyBank.viewBalance()
- [call] from: 0x58380a6a701c568545dcf803fc8875f56beddC4 to: MyBank.viewBalance() data: 0x3ff...1e05b