Name: Diya Theryani **Div:** D20B **Roll No.:** 66

Experiment No. 4

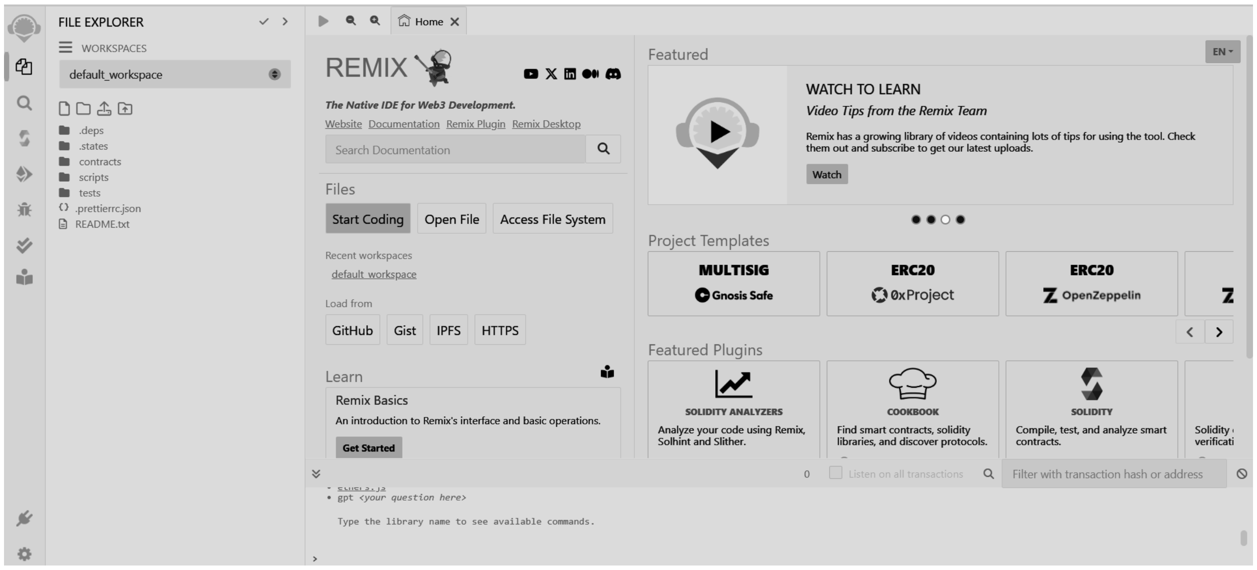
AIM: Create Smart Contract using Solidity and Remix IDE and Create Transactions using Solidity and Remix IDE

Tasks to be performed: (Write a Smart Contract on a test network for bank account of a customer for following operations: Deposit money, withdraw money and show balance by referring to the Remix-IDE tutorial)

1. Preparing Your Smart Contract Development Environment in Remix-IDE.
2. Creating Your Smart Contract File: Perform transactions among the peers.
3. Write the contract code.
4. Compile the contract.
5. Deploy smart contracts.
6. Interact with the deployed contract (testing).

Procedure-

1. Open [Remix IDE]() on the browser.



The Native IDE for Web3 Development.

Video Tips from the Remix Team

Remix has a growing library of videos containing lots of tips for using the tool. Check them out and subscribe to get our latest uploads.

An introduction to Remix's interface and basic operations.

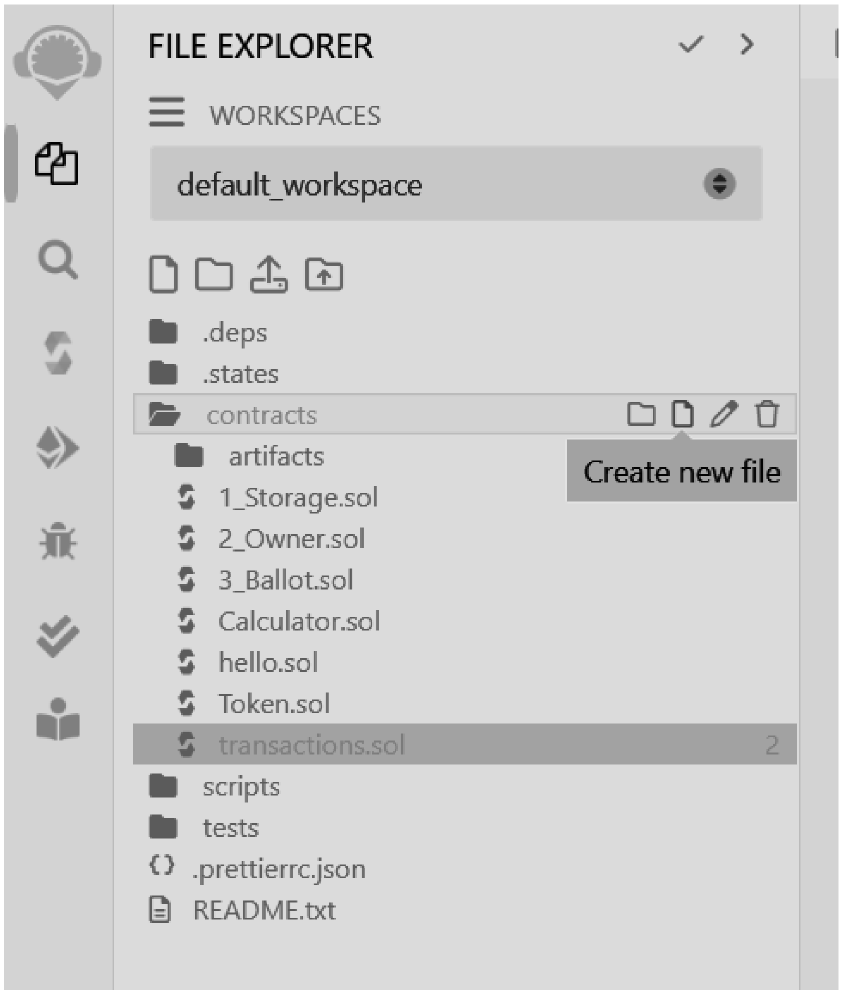
Analyze your code using Remix, Solhint and Slither.

Find smart contracts, solidity libraries, and discover protocols.

Compile, test, and analyze smart contracts.

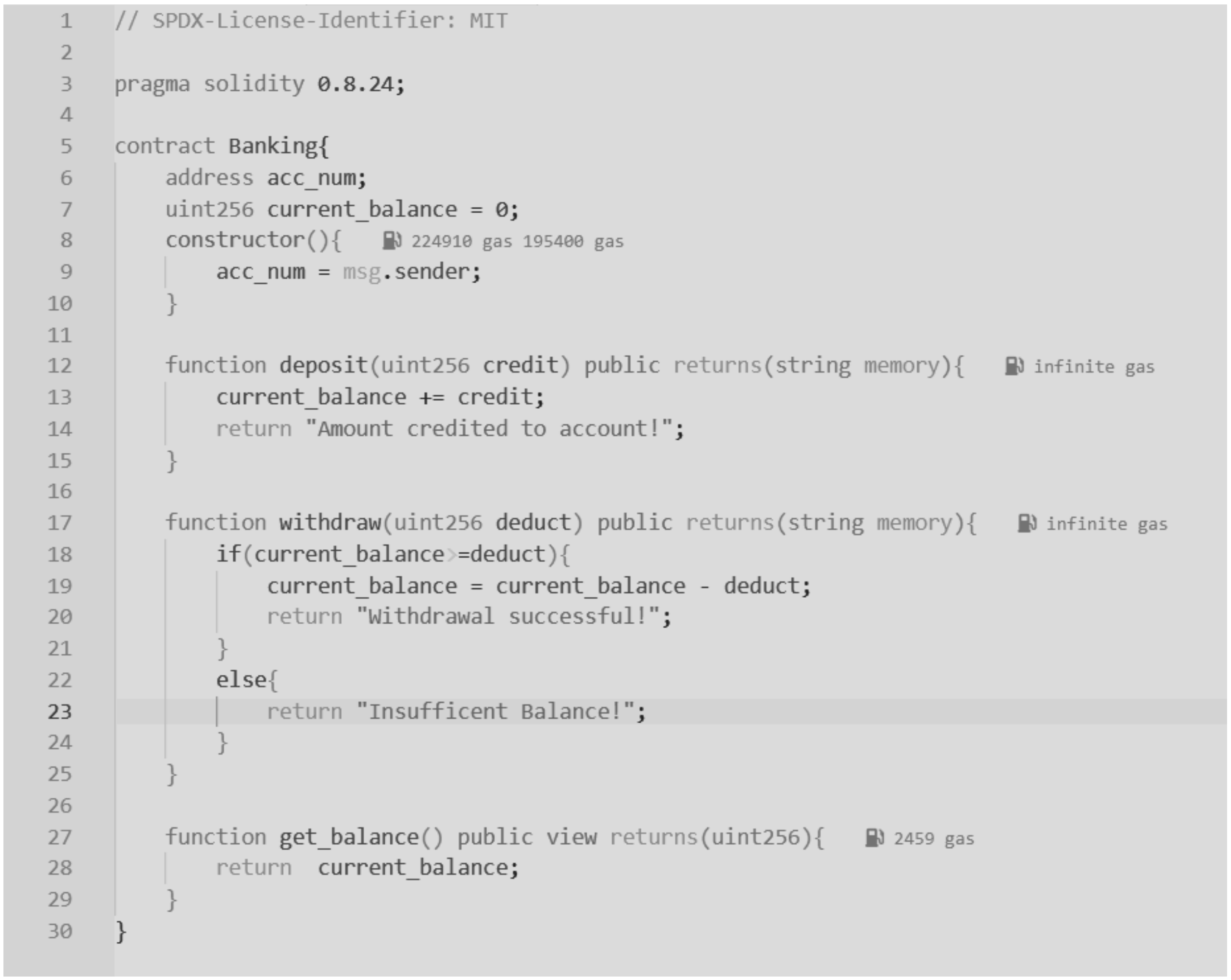
```
 • ethers.js
 • gpt <your question here>

 Type the library name to see available commands.
```

2. In the file explorer, click on the contracts folder and create a new file.



3. Write the code for the smart contract.

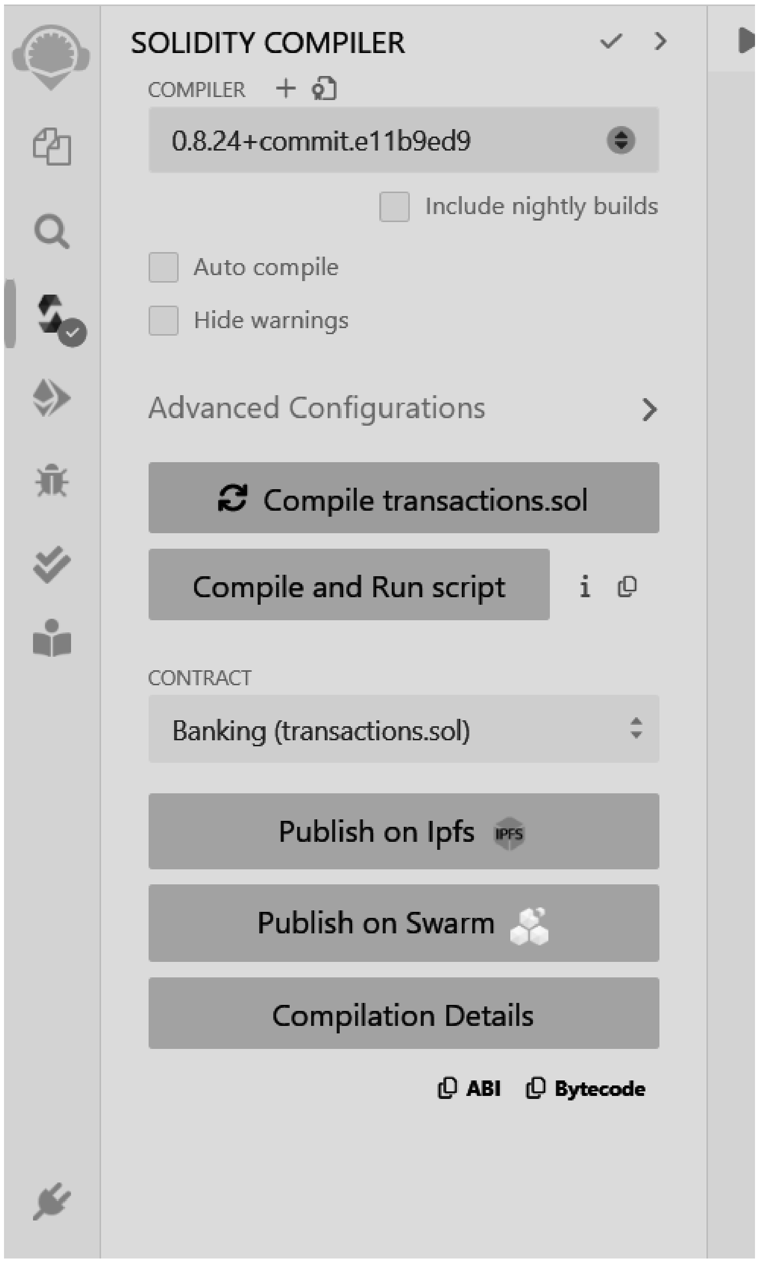
Name: Diya Theryani **Div:** D20B **Roll No.:** 66



```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.24;

contract Banking{
    address acc_num;
    uint256 current_balance = 0;
    constructor(){    224910 gas 195400 gas
        acc_num = msg.sender;
    }

    function deposit(uint256 credit) public returns(string memory){    infinite gas
        current_balance += credit;
        return "Amount credited to account!";
    }

    function withdraw(uint256 deduct) public returns(string memory){    infinite gas
        if(current_balance>=deduct){
            current_balance = current_balance - deduct;
            return "Withdrawal successful!";
        }
        else{
            return "Insufficent Balance!";
        }
    }

    function get_balance() public view returns(uint256){    2459 gas
        return  current_balance;
    }
}
```

Name: Diya Theryani **Div:** D20B **Roll No.:** 66

4. Compile the code by clicking on the 3rd option on the left hand side of the window and then click on the "Compile <filename>.sol" option.



SOLIDITY COMPILER

COMPILER

0.8.24+commit.e11b9ed9

Include nightly builds

Auto compile

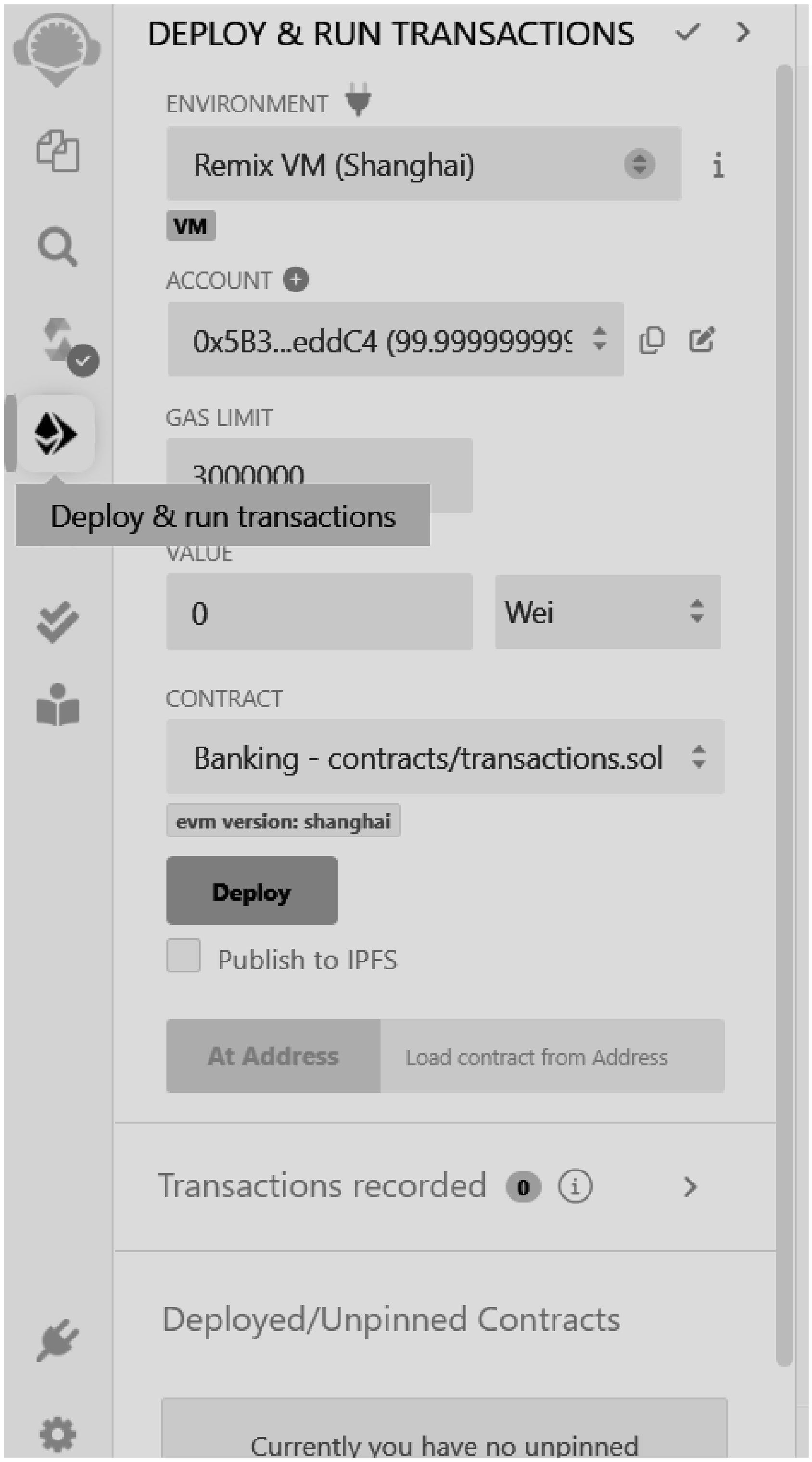
Hide warnings

CONTRACT

Banking (transactions.sol)

Name: Diya Theryani **Div:** D20B **Roll No.:** 66

5. Deploy the contract if compiled successfully by opening the “Deploy and Run”.



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

Remix VM (Shanghai)

VM

ACCOUNT

0x5B3...eddC4 (99.99999999

GAS LIMIT

3000000

Deploy & run transactions

VALUE

0

Wei

CONTRACT

Banking - contracts/transactions.sol

evm version: shanghai

Deploy

Publish to IPFS

At Address

Load contract from Address

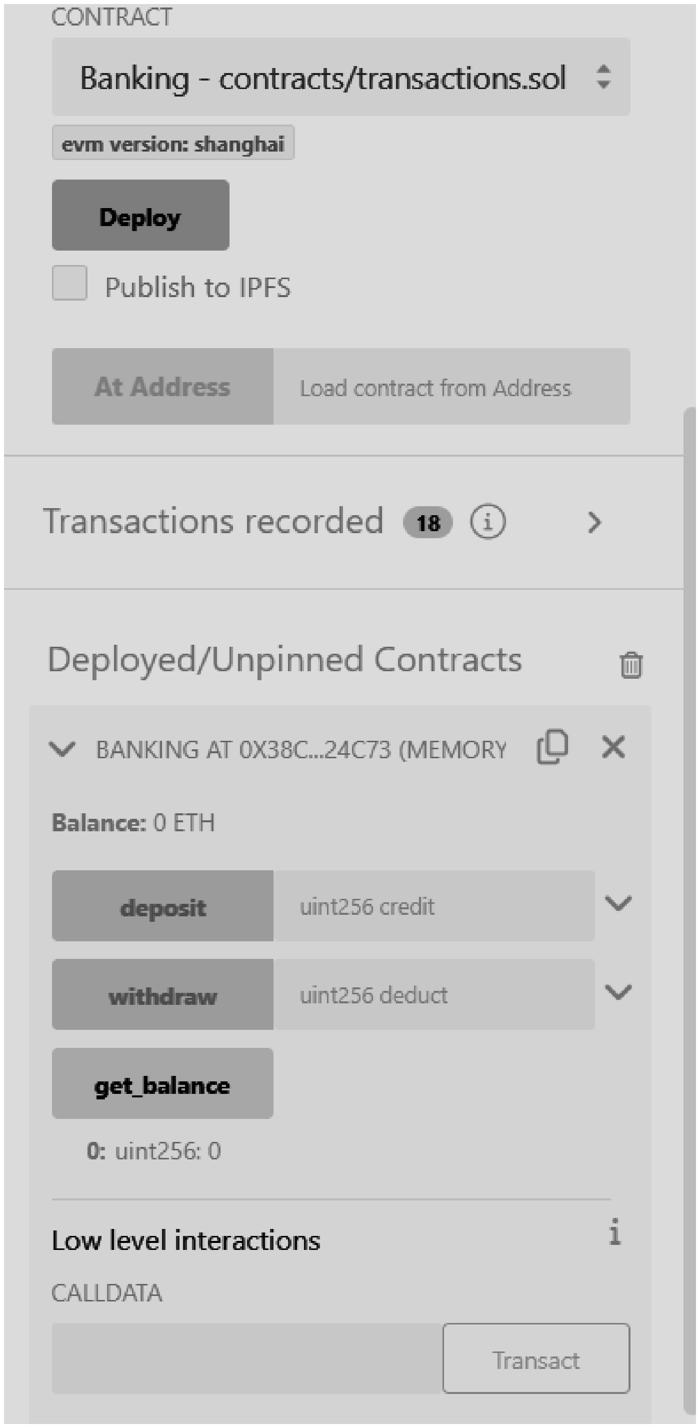
Transactions recorded 0

Deployed/Unpinned Contracts

Currently you have no unpinned

Name: Diya Theryani **Div:** D20B **Roll No.:** 66

6. Open the deployed contract where you will see the different functions we have written.



CONTRACT

Banking - contracts/transactions.sol

evm version: shanghai

Deploy

Publish to IPFS

At Address | Load contract from Address

Transactions recorded 18

Deployed/Unpinned Contracts

BANKING AT 0X38C...24C73 (MEMORY

Balance: 0 ETH

deposit | uint256 credit

withdraw | uint256 deduct

get_balance

0: uint256: 0

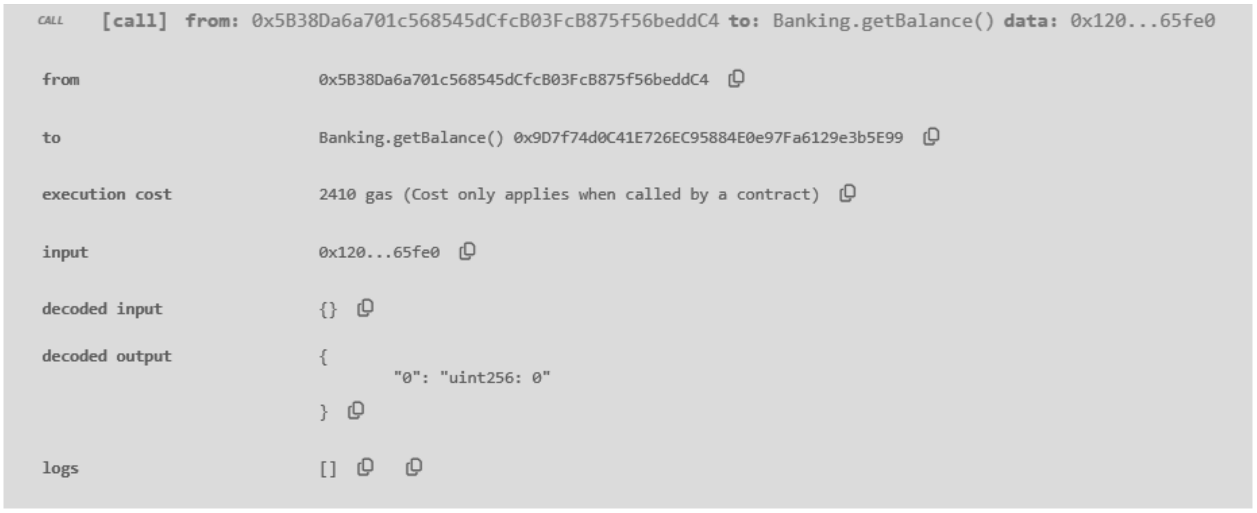
Low level interactions

CALLDATA

Transact

Name: Diya Theryani **Div:** D20B **Roll No.:** 66

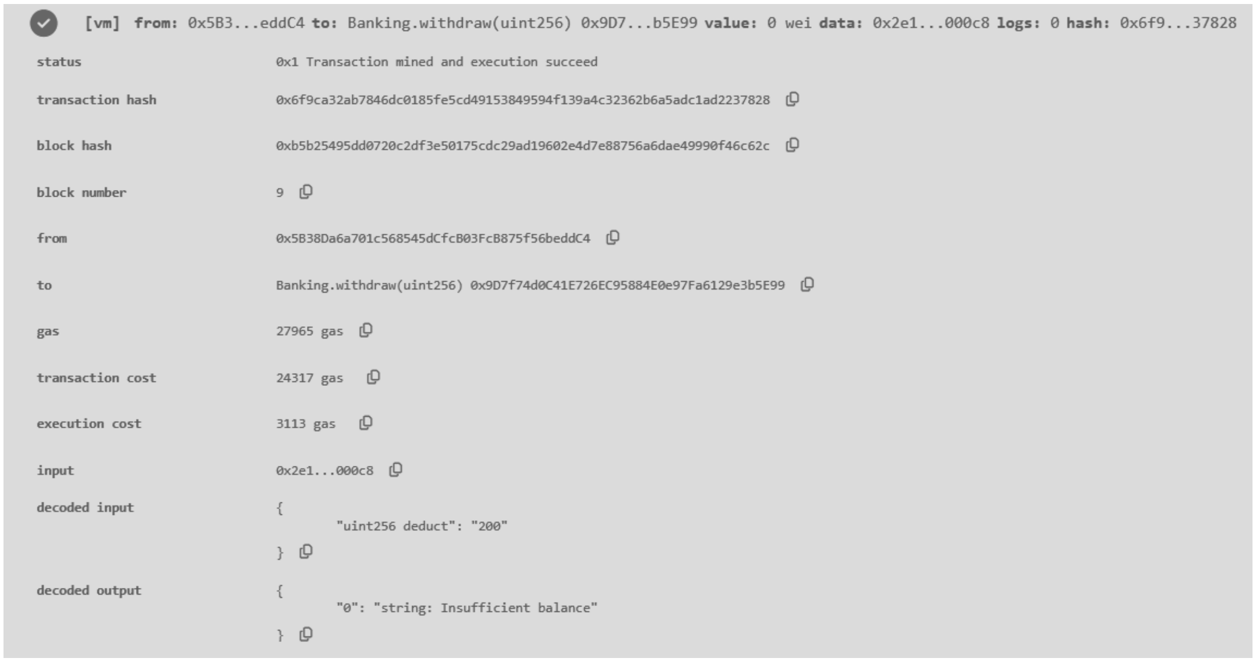
7. Balance-



```
CALL   [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Banking.getBalance() data: 0x120...65fe0

from                   0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to                     Banking.getBalance() 0x9D7f74d0C41E726EC95884E0e97Fa6129e3b5E99

execution cost         2410 gas (Cost only applies when called by a contract)

input                  0x120...65fe0

decoded input          {}

decoded output         {
                              "0": "uint256: 0"
                       }

logs                   []
```

Output message - "uint256: 0"

8. Withdraw(Low balance)-

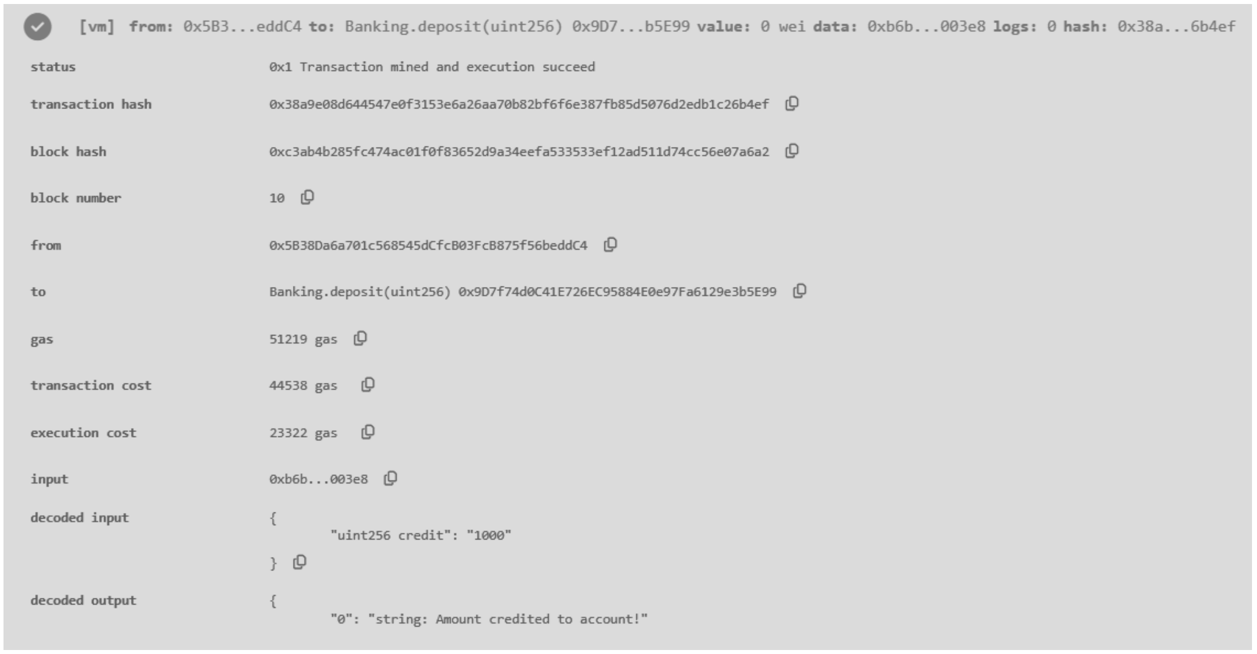


```
[vm]  from: 0x5B3...eddC4 to: Banking.withdraw(uint256) 0x9D7...b5E99 value: 0 wei data: 0x2e1...000c8 logs: 0 hash: 0x6f9...37828
status                 0x1 Transaction mined and execution succeed
transaction hash       0x6f9ca32ab7846dc0185fe5cd49153849594f139a4c32362b6a5adc1ad2237828
block hash             0xb5b25495dd0720c2df3e50175cdc29ad19602e4d7e88756a6dae49990f46c62c
block number           9
from                   0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                     Banking.withdraw(uint256) 0x9D7f74d0C41E726EC95884E0e97Fa6129e3b5E99
gas                    27965 gas
transaction cost       24317 gas
execution cost         3113 gas
input                  0x2e1...000c8
decoded input          {
                              "uint256 deduct": "200"
                       }
decoded output         {
                              "0": "string: Insufficient balance"
                       }
```

Output message - "string: Insufficient Balance!"

Name: Diya Theryani **Div:** D20B **Roll No.:** 66

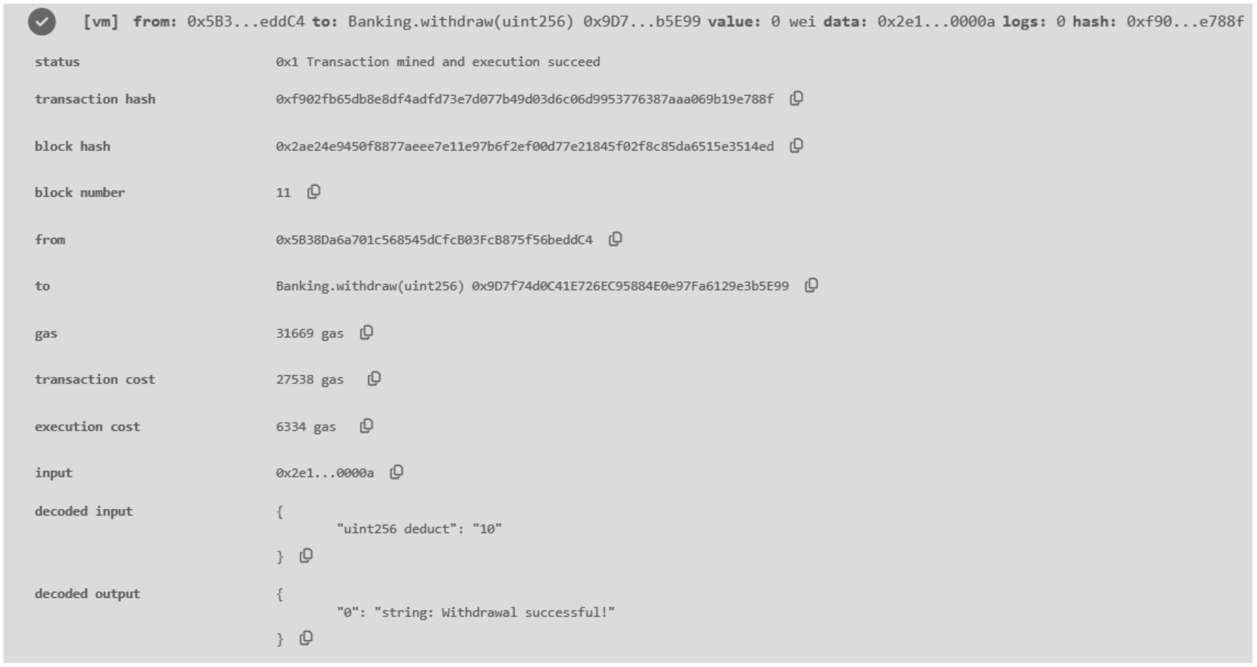
9. Deposit- (1000)



```
[vm] from: 0x5B3...eddC4 to: Banking.deposit(uint256) 0x9D7...b5E99 value: 0 wei data: 0xb6b...003e8 logs: 0 hash: 0x38a...6b4ef
status              0x1 Transaction mined and execution succeed
transaction hash    0x38a9e08d644547e0f3153e6a26aa70b82bf6f6e387fb85d5076d2edb1c26b4ef
block hash          0xc3ab4b285fc474ac01f0f83652d9a34eefa533533ef12ad511d74cc56e07a6a2
block number        10
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Banking.deposit(uint256) 0x9D7f74d0C41E726EC95884E0e97Fa6129e3b5E99
gas                 51219 gas
transaction cost    44538 gas
execution cost      23322 gas
input               0xb6b...003e8
decoded input       {
                           "uint256 credit": "1000"
                    }
decoded output      {
                           "0": "string: Amount credited to account!"
```

Output message - "string: Amount credited to account!"

10. Withdraw (10)-

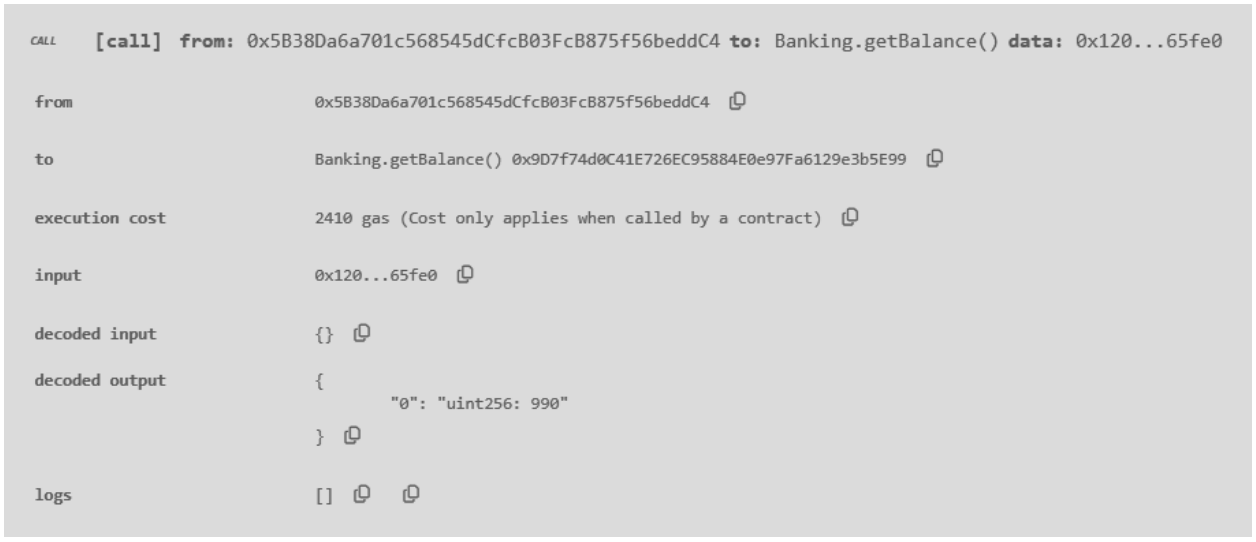


```
[vm] from: 0x5B3...eddC4 to: Banking.withdraw(uint256) 0x9D7...b5E99 value: 0 wei data: 0x2e1...0000a logs: 0 hash: 0xf90...e788f
status              0x1 Transaction mined and execution succeed
transaction hash    0xf902fb65db8e8df4adfd73e7d077b49d03d6c06d9953776387aaa069b19e788f
block hash          0x2ae24e9450f8877aeee7e11e97b6f2ef00d77e21845f02f8c85da6515e3514ed
block number        11
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Banking.withdraw(uint256) 0x9D7f74d0C41E726EC95884E0e97Fa6129e3b5E99
gas                 31669 gas
transaction cost    27538 gas
execution cost      6334 gas
input               0x2e1...0000a
decoded input       {
                           "uint256 deduct": "10"
                    }
decoded output      {
                           "0": "string: Withdrawal successful!"
                    }
```

Output message - "string: Withdrawal successful!"

Name: Diya Theryani **Div:** D20B **Roll No.:** 66

11. Balance -



```
CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Banking.getBalance() data: 0x120...65fe0

from                 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                   Banking.getBalance() 0x9D7f74d0C41E726EC95884E0e97Fa6129e3b5E99
execution cost       2410 gas (Cost only applies when called by a contract)
input                0x120...65fe0
decoded input        {}
decoded output       {
                         "0": "uint256: 990"
                     }
logs                 []
```

Output message - "uint256: 990"

Conclusion:
In this experiment, we created a smart contract on remix ide using solidity language and performed transactions like withdraw and show balance from remix ide.