BLOCKCHAIN TECHNOLOGY LAB

(20CP406P)

LAB ASSIGNMENT - 7

B.Tech in Computer Science and Engineering Dept.,

Pandit Deendayal Energy University,

Gandhinagar

Name: Vrushank Ariwala

Roll No.: 19BCP141

Branch: CSE

❖ **Aim:**

Create the Banking Application, Deploy it on Testnet through Metamask.

❖ **Banking Smart Contract:**

Implementation:

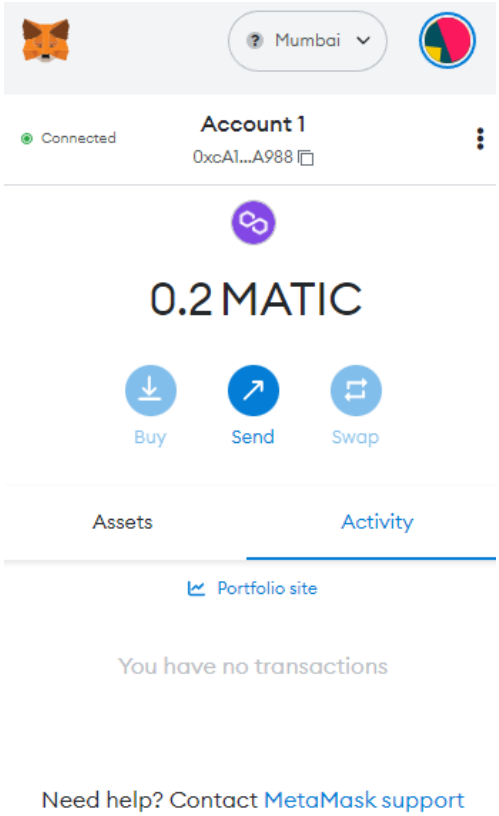
Step1: Write your Smart Contract in Solidity IDE

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract banking{
  mapping(address=>uint) public userAccount;
  mapping(address=>bool) public userExists;

  function createAcc() public payable returns(string memory){
      require(userExists[msg.sender]==false,'Account Already
Created');
      if(msg.value==0){
          userAccount[msg.sender]=0;
          userExists[msg.sender]=true;
          return 'account created';
      }
      require(userExists[msg.sender]==false,'account already
created');
      userAccount[msg.sender] = msg.value;
      userExists[msg.sender] = true;
      return 'account created';
  }

  function deposit(uint amount) public payable returns(string
memory){
      require(userExists[msg.sender]==true, 'Account is not
created');
      require(amount>0, 'Value for deposit is Zero');
      userAccount[msg.sender]=userAccount[msg.sender]+amount;
      return 'Deposited Succesfully';
  }

  function withdraw(uint amount) public payable returns(string
memory){
      require(userAccount[msg.sender]>amount, 'insufficeint balance
in Bank account');
```

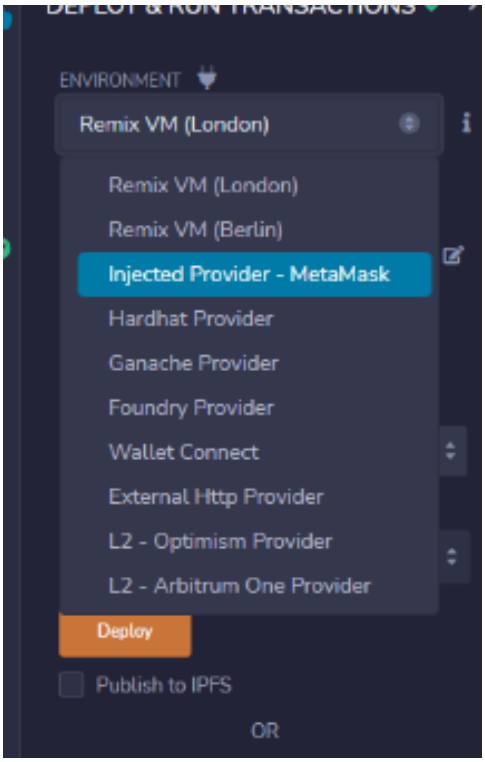
```
        require(userExists[msg.sender]==true, 'Account is not
created');
        require(amount>0, 'Enter non-zero value for withdrawal');
        userAccount[msg.sender]=userAccount[msg.sender]-amount;
        payable(msg.sender).transfer(amount);
        return 'withdrawal Succesful';
    }

    function TransferAmount(address payable userAddress, uint amount)
public returns(string memory){
        require(userAccount[msg.sender]>amount, 'insufficeint balance
in Bank account');
        require(userExists[msg.sender]==true, 'Account is not
created');
        require(userExists[userAddress]==true, 'to Transfer account
does not exists in bank accounts ');
        require(amount>0, 'Enter non-zero value for sending');
        userAccount[msg.sender]=userAccount[msg.sender]-amount;
        userAccount[userAddress]=userAccount[userAddress]+amount;
        return 'transfer succesfully';
    }

    function sendAmount(address payable toAddress , uint256 amount)
public payable returns(string memory){
        require(amount>0, 'Enter non-zero value for withdrawal');
        require(userExists[msg.sender]==true, 'Account is not
created');
        require(userAccount[msg.sender]>amount, 'insufficeint balance
in Bank account');
        userAccount[msg.sender]=userAccount[msg.sender]-amount;
        toAddress.transfer(amount);
        return 'transfer success';
    }

    function userAccountBalance() public view returns(uint){
        return userAccount[msg.sender];
    }

    function accountExist() public view returns(bool){
        return userExists[msg.sender];
    }

}
```

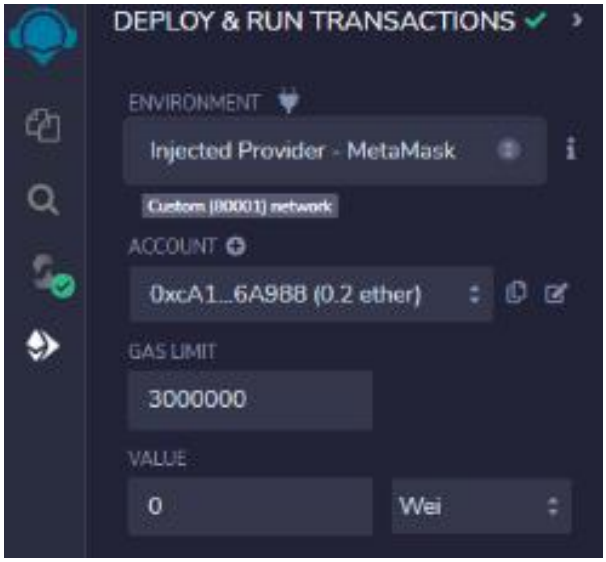
Step2: Open Metamask Extension.

I am using Mumbai Testnet.

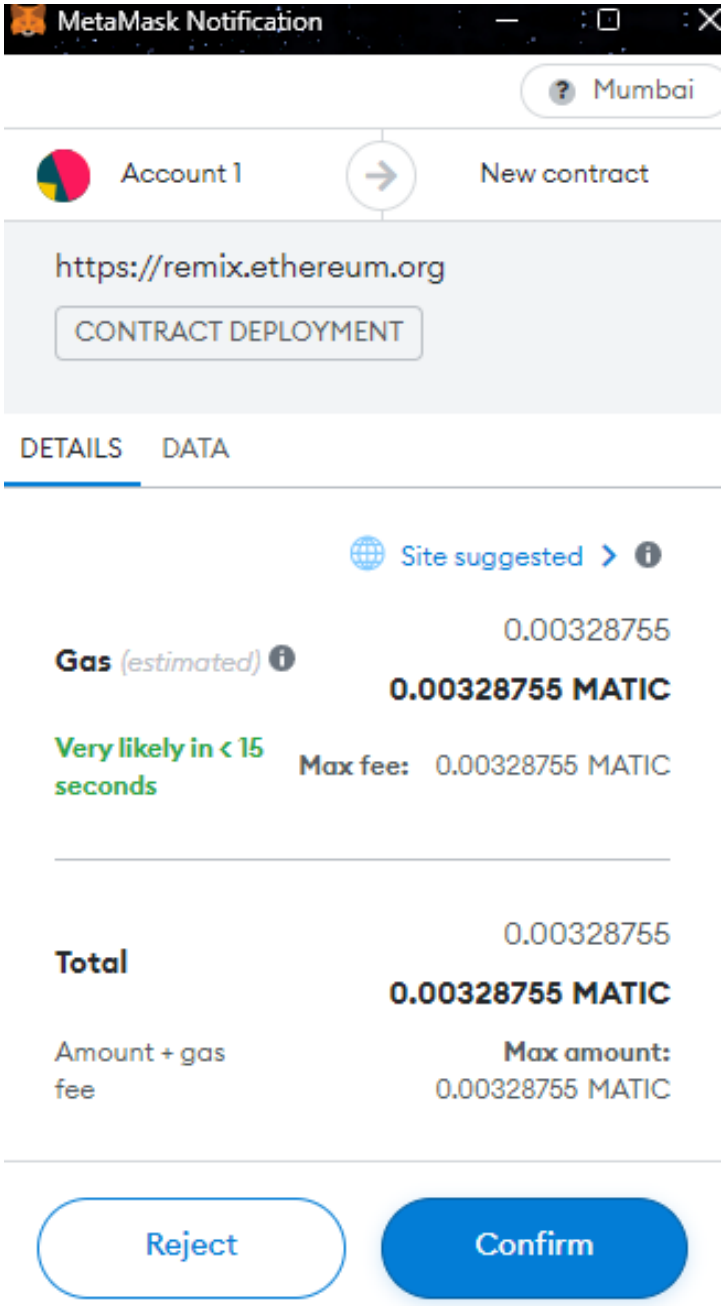


Step3: In compiler Select Environment to ‘Injected Provider Metamask’.





Step4: Now deploy your Smart Contract. Then confirm it in Metamask.



Step5: Your Smart Contract is Confirmed.

Contract deployment

Status View on block explorer

Confirmed Copy transaction ID

From	To
0xcA1...A988	New contract

Transaction

Nonce	0
Amount	**-0 MATIC**
Gas Limit (Units)	1315021
Gas Used (Units)	1315021
Base fee (GWEI)	0.000000008
Priority fee (GWEI)	2.5
Total gas fee	0.003288 MATIC
Max fee per gas	0.000000003 MATIC
Total	**0.00328755 MATIC**