Centurion UNIVERSITY
Shaping Lives... Empowering Communities...

School: .... Campus: ....

Academic Year: .... Subject Name: .... Subject Code: ....

Semester: .... Program: .... Branch: .... Specialization: ....

Date: ....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement :

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ALGORITHM:

1.Start by creating two ERC-20 tokens (TokenA and TokenB) using Solidity.
2.Deploy both token contracts on the test network (e.g., Sepolia or Goerli).
3.Note down the deployed contract addresses for both tokens.
4.Open MetaMask and import both token addresses to display balances.
5.Write the AMM (Automated Market Maker) smart contract in Solidity.
6.Implement functions for adding liquidity, removing liquidity, and swapping tokens.
7.Compile the AMM smart contract without errors.
8.Deploy the AMM contract on the same network as your tokens.
9.Copy and save the AMM contract address for further steps.
10.From MetaMask, approve the AMM contract to spend a chosen amount of TokenA.
11.Similarly, approve the AMM contract to spend a chosen amount of TokenB.
12.Call the AMM contract's addLiquidity function to deposit TokenA and TokenB into the pool.
13.Verify that the liquidity has been added successfully by checking reserves.
14.Call the AMM contract's swap function to exchange one token for another.
15.Confirm the swap transaction and check MetaMask to ensure balances are updated.

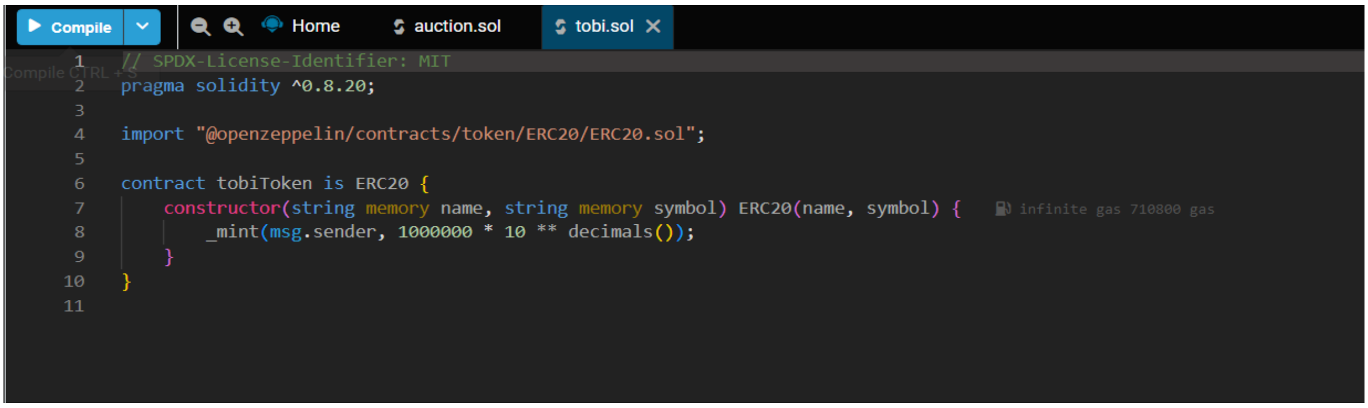
* Softwares used

1.Remix IDE
2.MetaMask Wallet
3.Brave Web Browser
4.Ethereum Test Network -Sepolia.
5.Etherscan Testnet Explorer

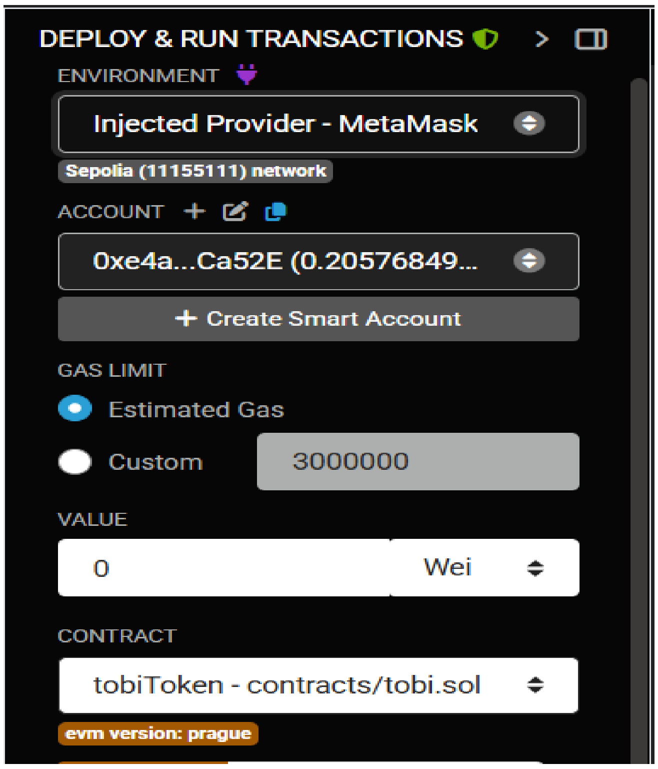
***** *As applicable according to the experiment. Two sheets per experiment (10-20) to be used.*

* Testing Phase: Compilation of Code (error detection)

First create your two tokens using ERC20 i have already created two token one is TOBI token and another is MAD token and i already import them in my metamask wallet .This is the smart contract for creating your own token ,after compiling the smart contract in deploy time we have to pass the string token name and symbol of our token (e.g-tobiToken,TOBI) after contract deploy go to metamask and explore the transaction on eterscan and copy the contract address of the token and in metamsk tokens section click on import tokens in this we have to give the testnet network we used (e.g-sepolia) and patse the contract address then you see our token is successfully added to our metamask wallet.



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract tobiToken is ERC20 {
    constructor(string memory name, string memory symbol) ERC20(name, symbol) {    infinite gas 710800 gas
        _mint(msg.sender, 1000000 * 10 ** decimals());
    }
}
```


DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask

Sepolia (11155111) network

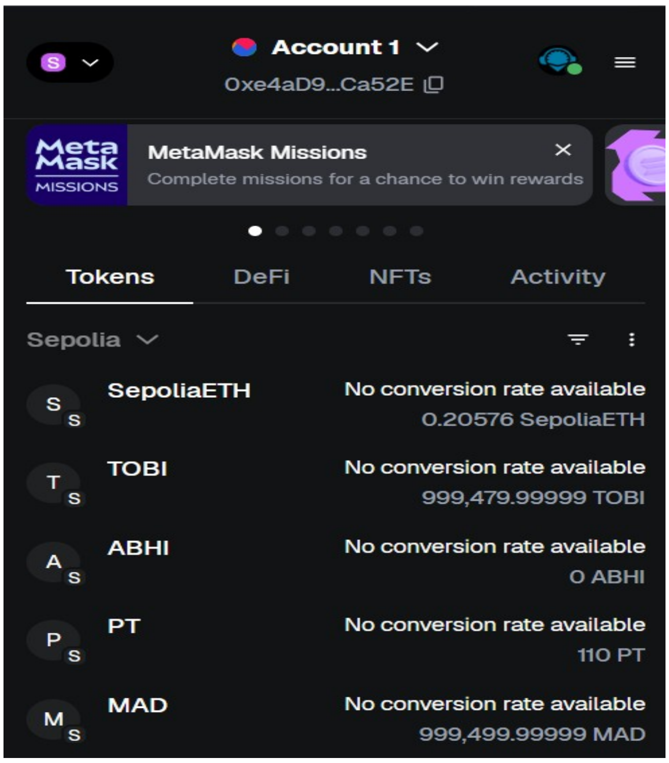
ACCOUNT: 0xe4a...Ca52E (0.20576849...

GAS LIMIT: [✓ Estimated Gas] Custom 3000000

VALUE: 0 Wei

CONTRACT: tobiToken - contracts/tobi.sol

evm version: prague



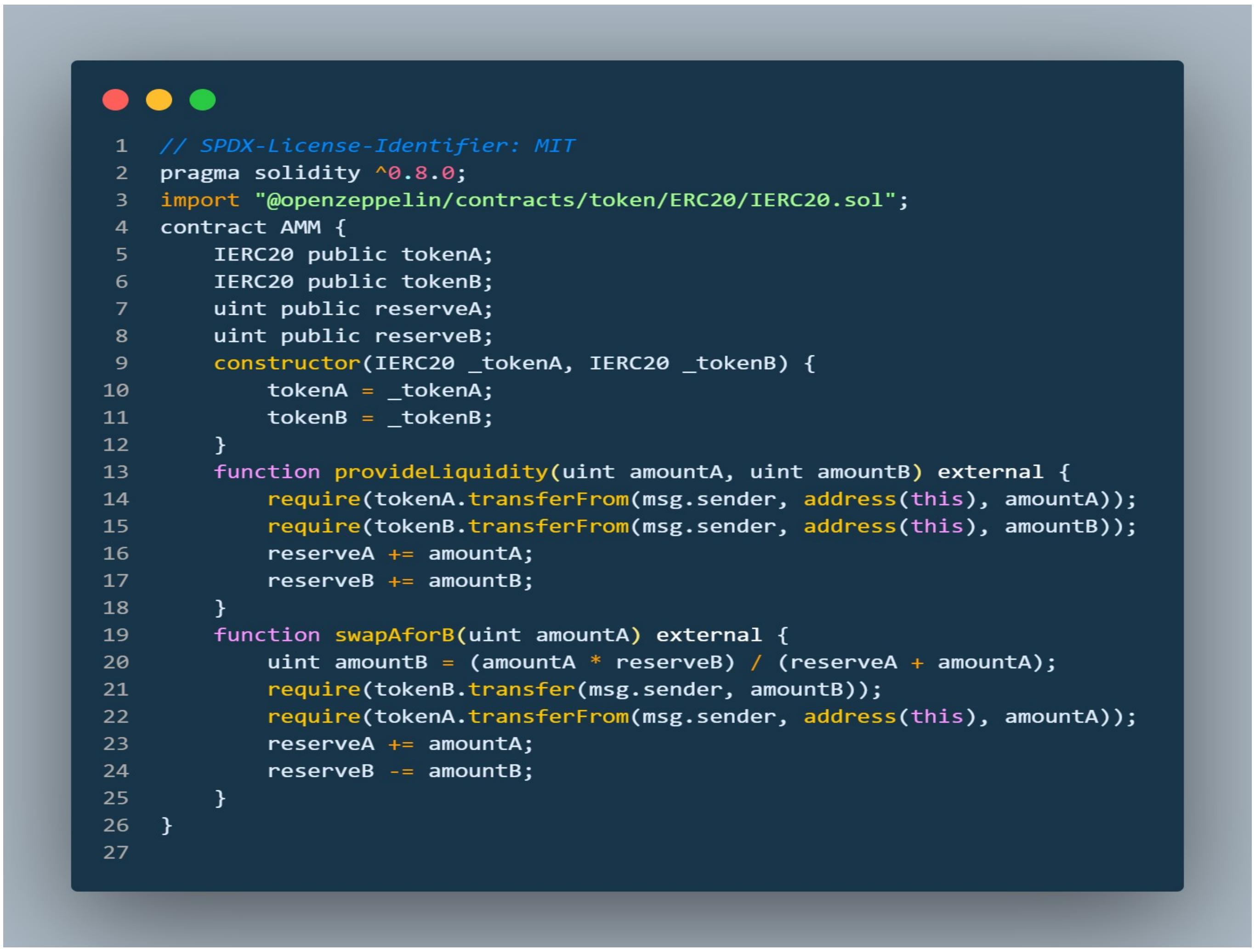
SepoliaETH	No conversion rate available 0.20576 SepoliaETH
TOBI	No conversion rate available 999,479.99999 TOBI
ABHI	No conversion rate available 0 ABHI
PT	No conversion rate available 110 PT
MAD	No conversion rate available 999,499.99999 MAD

Now we can see the token has successfully added to our metamask wallet . Now we have to write a smart contract for addliquidity and swap function

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

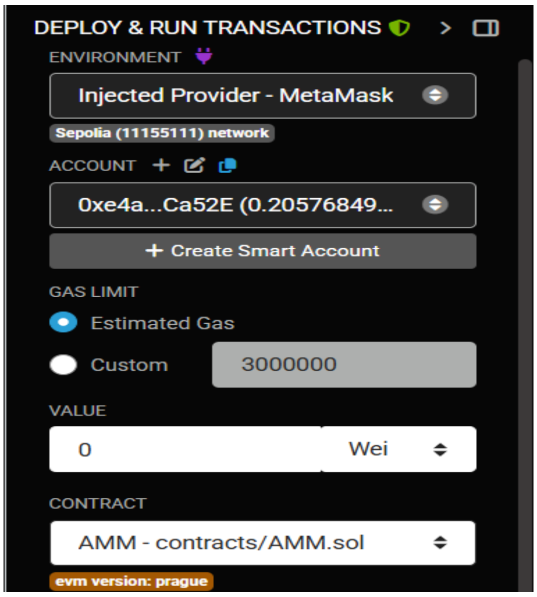
* Testing Phase: Compilation of Code (error detection)

The smart contract for AMM is including functions like providesolidity and swapforAandB .



```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
contract AMM {
    IERC20 public tokenA;
    IERC20 public tokenB;
    uint public reserveA;
    uint public reserveB;
    constructor(IERC20 _tokenA, IERC20 _tokenB) {
        tokenA = _tokenA;
        tokenB = _tokenB;
    }
    function provideLiquidity(uint amountA, uint amountB) external {
        require(tokenA.transferFrom(msg.sender, address(this), amountA));
        require(tokenB.transferFrom(msg.sender, address(this), amountB));
        reserveA += amountA;
        reserveB += amountB;
    }
    function swapAforB(uint amountA) external {
        uint amountB = (amountA * reserveB) / (reserveA + amountA);
        require(tokenB.transfer(msg.sender, amountB));
        require(tokenA.transferFrom(msg.sender, address(this), amountA));
        reserveA += amountA;
        reserveB -= amountB;
    }
}
```

Now compile the smart contract without any error after successfully compilation we have to deploy the smart contract before deploy the smart contract first we have to choose the injector provider as metamask



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask

Sepolia (11155111) network

ACCOUNT: 0xe4a...Ca52E (0.20576849...

+ Create Smart Account

GAS LIMIT: ● Estimated Gas ○ Custom 3000000

VALUE: 0 Wei

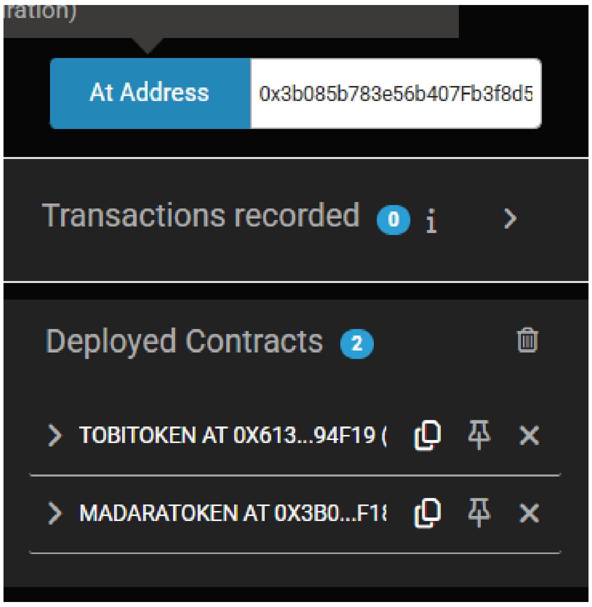
CONTRACT: AMM - contracts/AMM.sol

evm version: prague

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

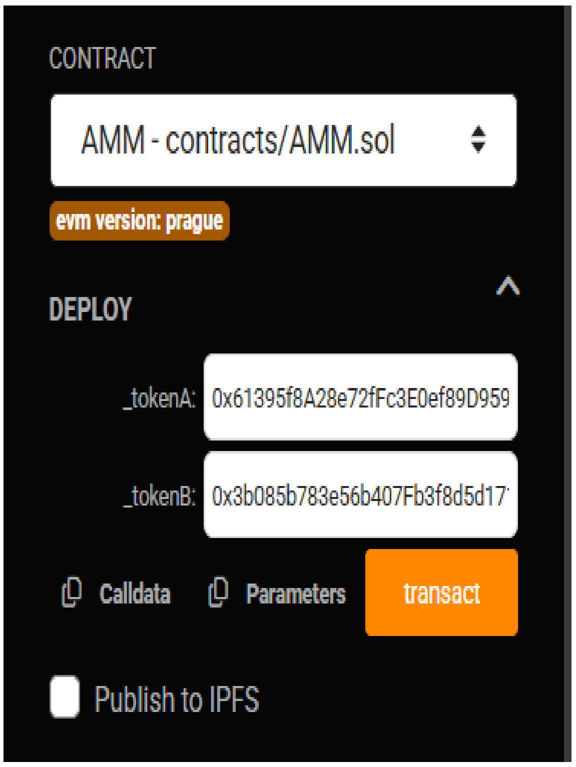
* Testing Phase: Compilation of Code (error detection)

Now add two previous deployed ERC20 tokens in the deployed section

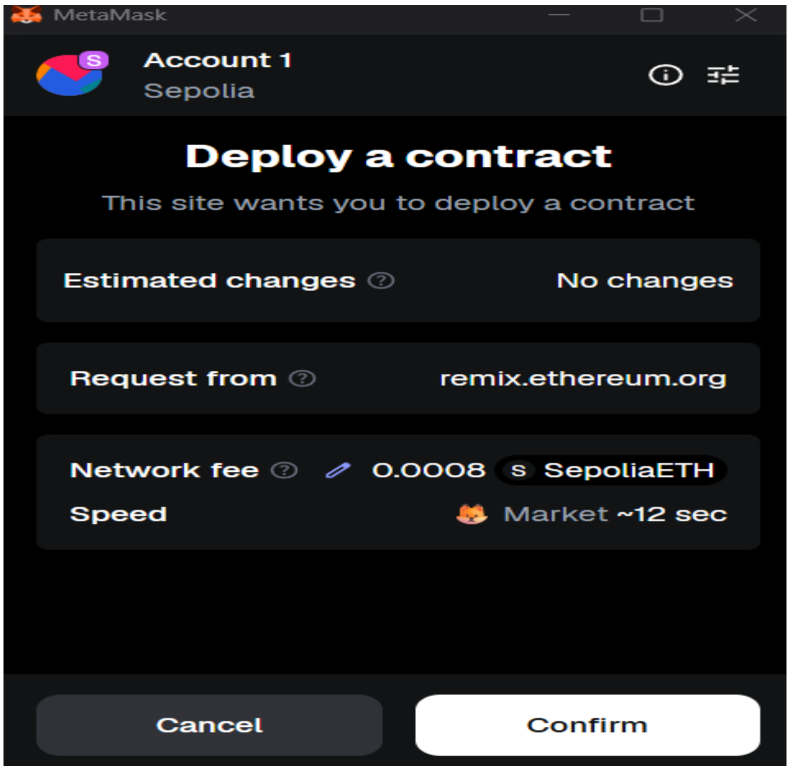


```
ration)
At Address   0x3b085b783e56b407Fb3f8d5
Transactions recorded 0
Deployed Contracts 2
> TOBITOKEN AT 0X613...94F19 (
> MADARATOKEN AT 0X3B0...F1
```

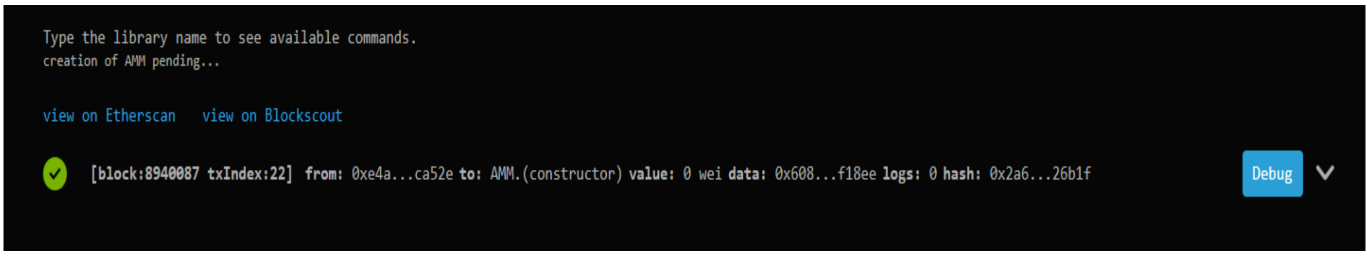
Now Deploy the AMM smart contract to deploy the smart contract we have to give the contract address of two tokens



```
CONTRACT
AMM - contracts/AMM.sol
evm version: prague
DEPLOY
_tokenA: 0x61395f8A28e72fFc3E0ef89D959
_tokenB: 0x3b085b783e56b407Fb3f8d5d17
Calldata   Parameters   transact
Publish to IPFS
```



```
MetaMask
Account 1
Sepolia
Deploy a contract
This site wants you to deploy a contract
Estimated changes          No changes
Request from               remix.ethereum.org
Network fee                0.0008 SepoliaETH
Speed                      Market ~12 sec
Cancel                     Confirm
```

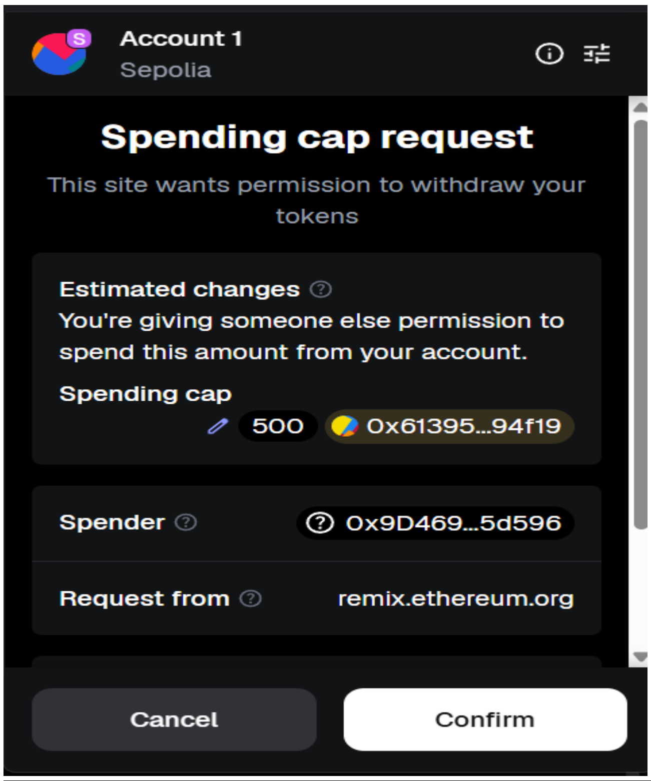


```
Type the library name to see available commands.
creation of AMM pending...

view on Etherscan   view on Blockscout

[block:8940087 txIndex:22] from: 0xe4a...ca52e to: AMM.(constructor) value: 0 wei data: 0x608...f18ee logs: 0 hash: 0x2a6...26b1f
```

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Testing Phase: Compilation of Code (error detection)

Now we have give access to the tokens for swapping and provideliquidity. For giving access copy the contract address of AMM and in tobiToken contract in approve function we have to pass AMM contract address and give some uint value for transaction and so same for transfer function.



Account 1
Sepolia

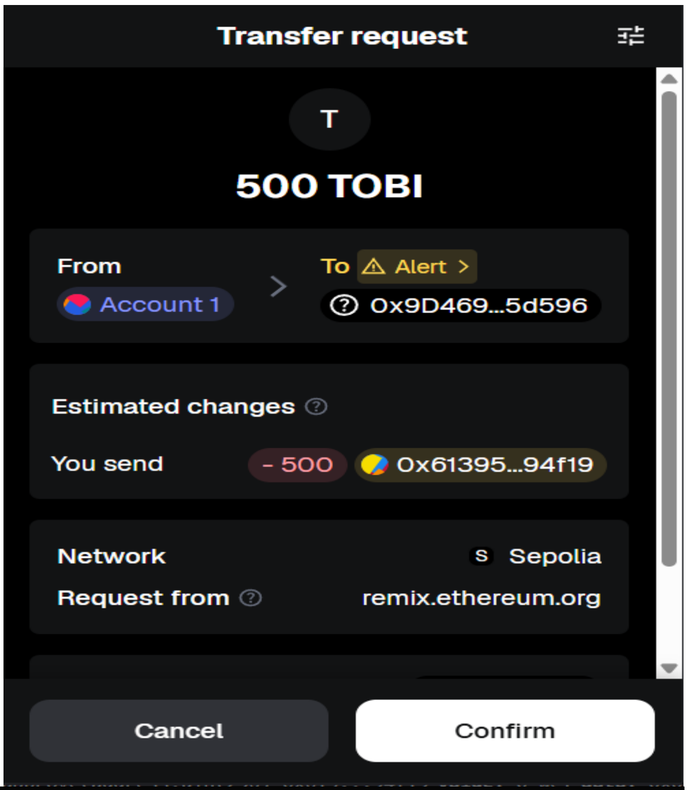
Spending cap request

This site wants permission to withdraw your tokens

Estimated changes
You're giving someone else permission to spend this amount from your account.
Spending cap 500 0x61395...94f19

Spender 0x9D469...5d596

Request from remix.ethereum.org



Transfer request

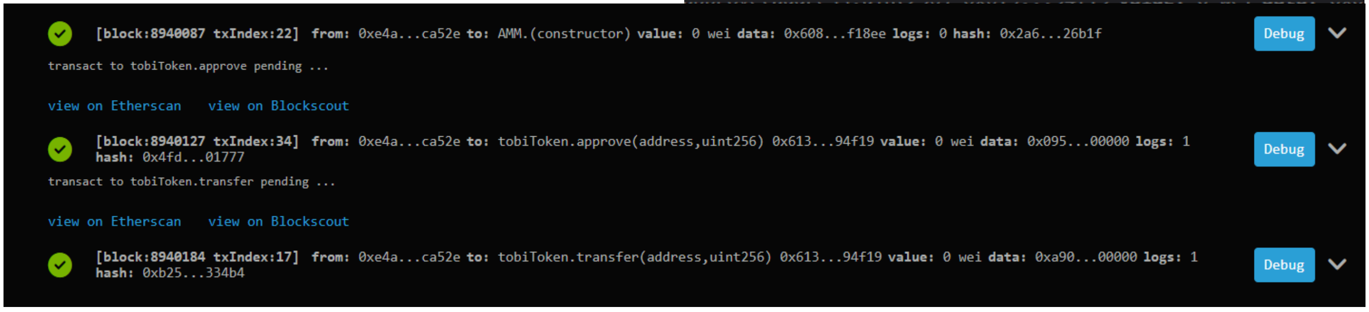
T

500 TOBI

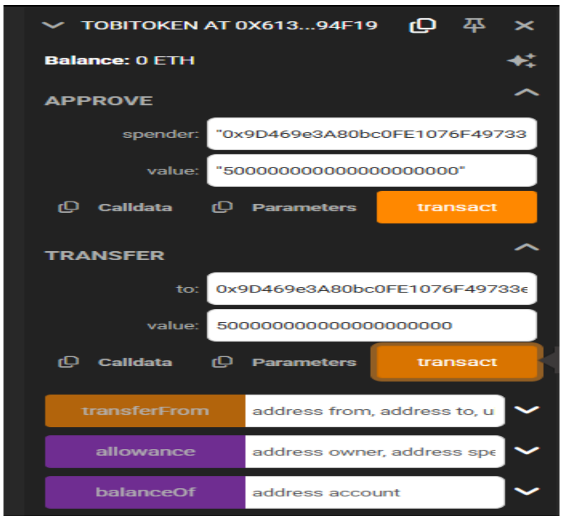
From Account 1
To ⚠ Alert > 0x9D469...5d596

Estimated changes
You send - 500 0x61395...94f19

Network S Sepolia
Request from remix.ethereum.org



```
[block:8940087 txIndex:22] from: 0xe4a...ca52e to: AMM.(constructor) value: 0 wei data: 0x608...f18ee logs: 0 hash: 0x2a6...26b1f
transact to tobiToken.approve pending ...

view on Etherscan   view on Blockscout
[block:8940127 txIndex:34] from: 0xe4a...ca52e to: tobiToken.approve(address,uint256) 0x613...94f19 value: 0 wei data: 0x095...00000 logs: 1
hash: 0x4fd...01777
transact to tobiToken.transfer pending ...

view on Etherscan   view on Blockscout
[block:8940184 txIndex:17] from: 0xe4a...ca52e to: tobiToken.transfer(address,uint256) 0x613...94f19 value: 0 wei data: 0xa90...00000 logs: 1
hash: 0xb25...334b4
```



TOBITOKEN AT 0X613...94F19

Balance: 0 ETH

APPROVE
spender: "0x9D469e3A80bc0FE1076F49733
value: "500000000000000000000"

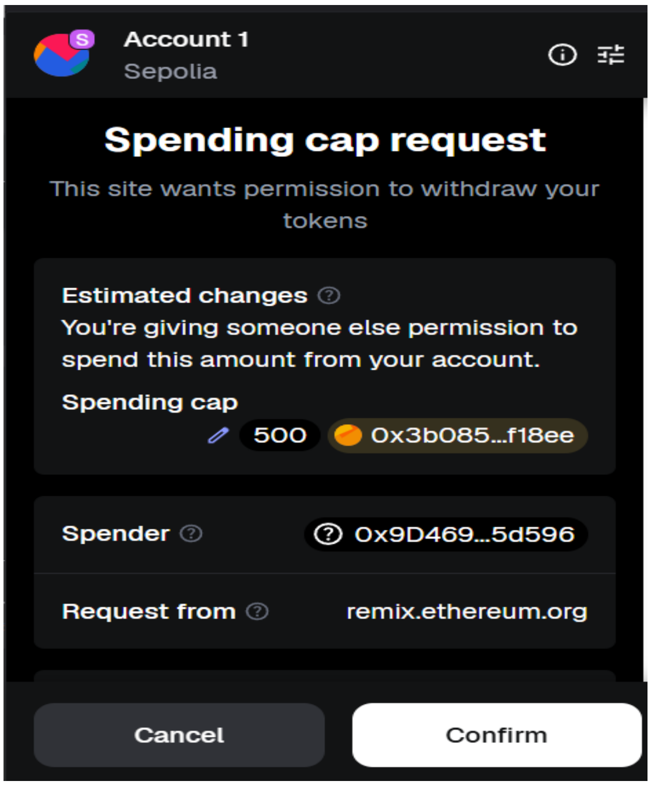
TRANSFER
to: 0x9D469e3A80bc0FE1076F49733e
value: 500000000000000000000

transferFrom: address from, address to, u
allowance: address owner, address spe
balanceOf: address account

Now do the smae steps for the another token MAD.

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Testing Phase: Compilation of Code (error detection)



Spending cap request

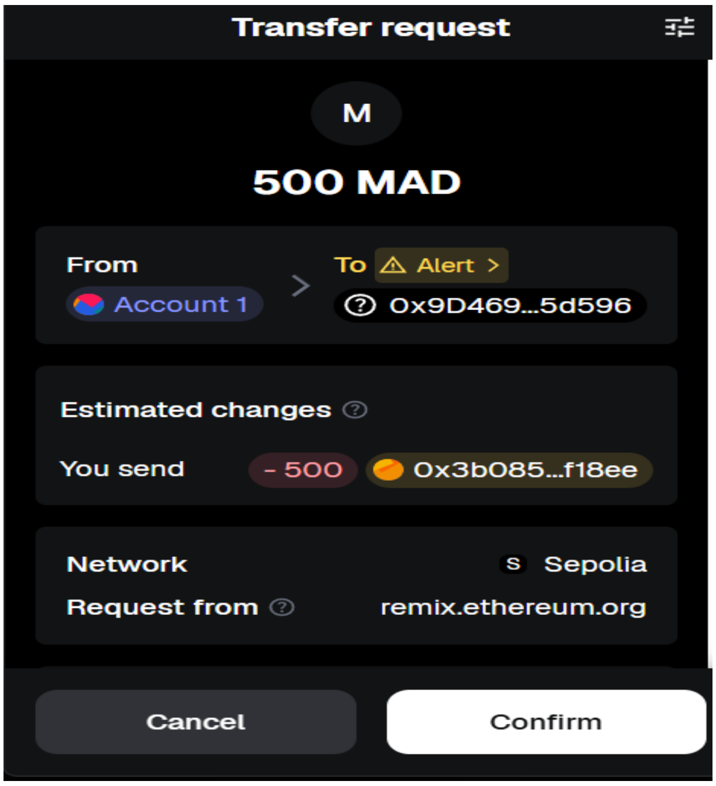
This site wants permission to withdraw your tokens

Estimated changes
You're giving someone else permission to spend this amount from your account.

Spending cap
500 0x3b085...f18ee

Spender 0x9D469...5d596

Request from remix.ethereum.org



Transfer request

M

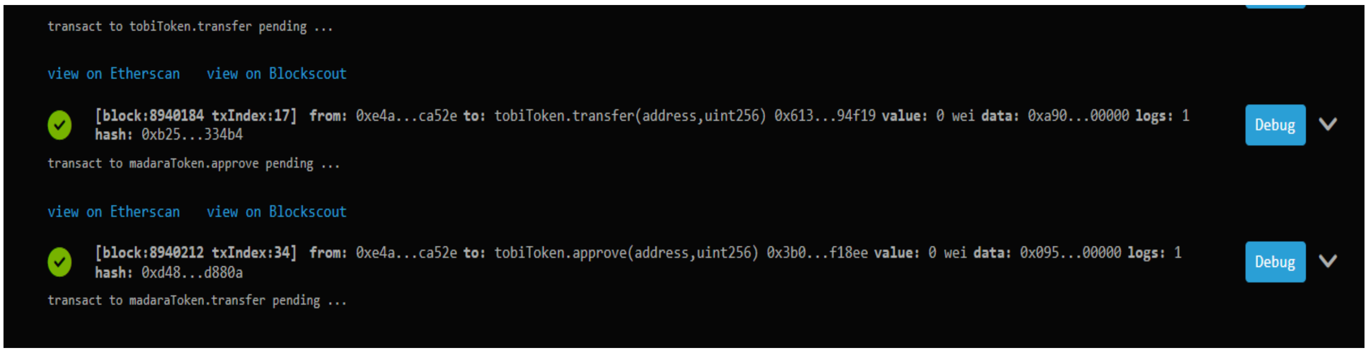
500 MAD

From Account 1
To Alert > 0x9D469...5d596

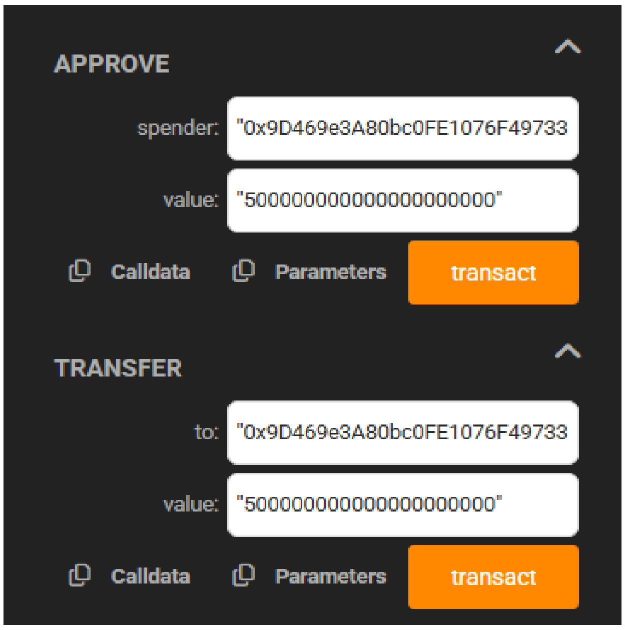
Estimated changes

You send - 500 0x3b085...f18ee

Network Sepolia
Request from remix.ethereum.org



```
transact to tobiToken.transfer pending ...

view on Etherscan   view on Blockscout

[block:8940184 txIndex:17] from: 0xe4a...ca52e to: tobiToken.transfer(address,uint256) 0x613...94f19 value: 0 wei data: 0xa90...00000 logs: 1
hash: 0xb25...334b4
transact to madaraToken.approve pending ...

view on Etherscan   view on Blockscout

[block:8940212 txIndex:34] from: 0xe4a...ca52e to: tobiToken.approve(address,uint256) 0x3b0...f18ee value: 0 wei data: 0x095...00000 logs: 1
hash: 0xd48...d880a
transact to madaraToken.transfer pending ...
```



APPROVE

spender: "0x9D469e3A80bc0FE1076F49733

value: "500000000000000000000"

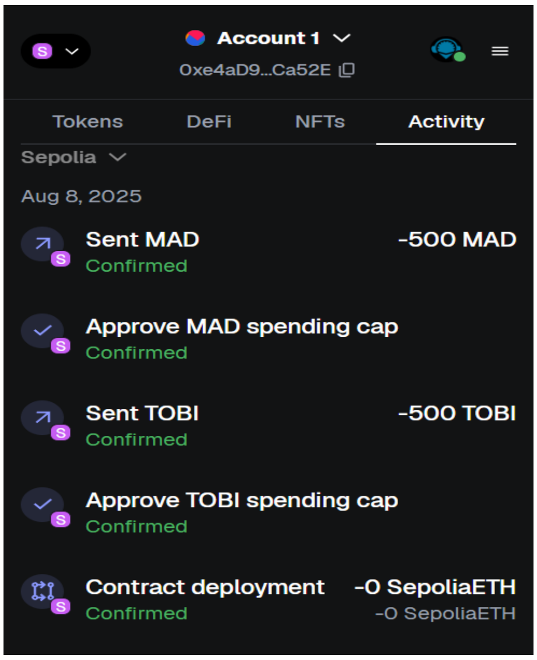
TRANSFER

to: "0x9D469e3A80bc0FE1076F49733

value: "500000000000000000000"

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Testing Phase: Compilation of Code (error detection)



Account 1
0xe4aD9...Ca52E

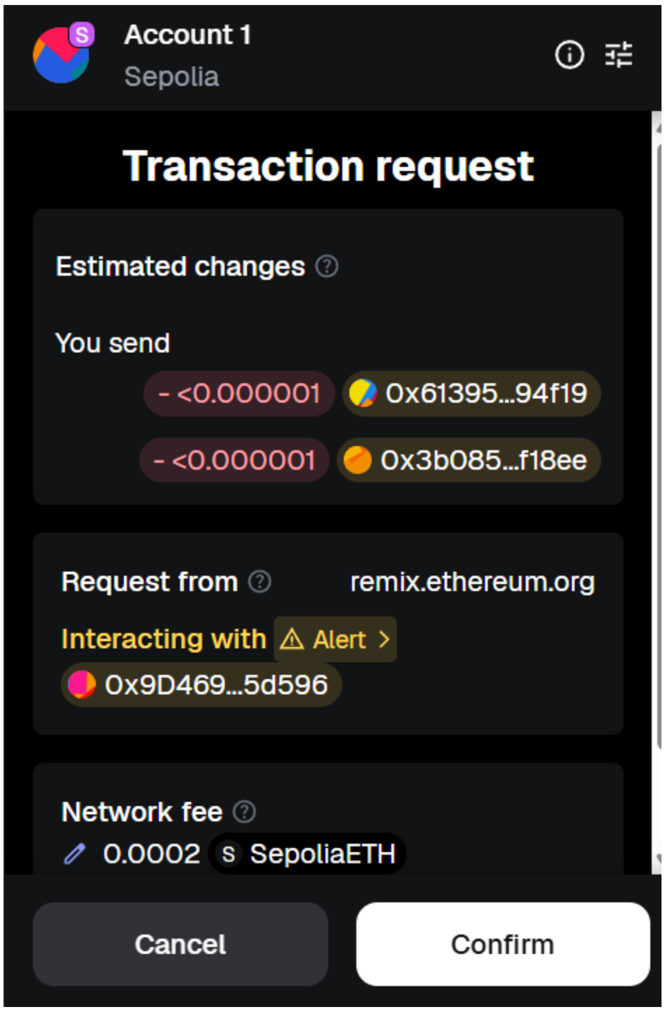
Tokens DeFi NFTs Activity

Sepolia

Aug 8, 2025

```
Sent MAD                              -500 MAD
Confirmed
Approve MAD spending cap
Confirmed
Sent TOBI                             -500 TOBI
Confirmed
Approve TOBI spending cap
Confirmed
Contract deployment          -0 SepoliaETH
Confirmed                    -0 SepoliaETH
```

Now after giving access to the token now time to check the provide liquidity to check liquidity give amountA and amountB



Account 1
Sepolia

Transaction request

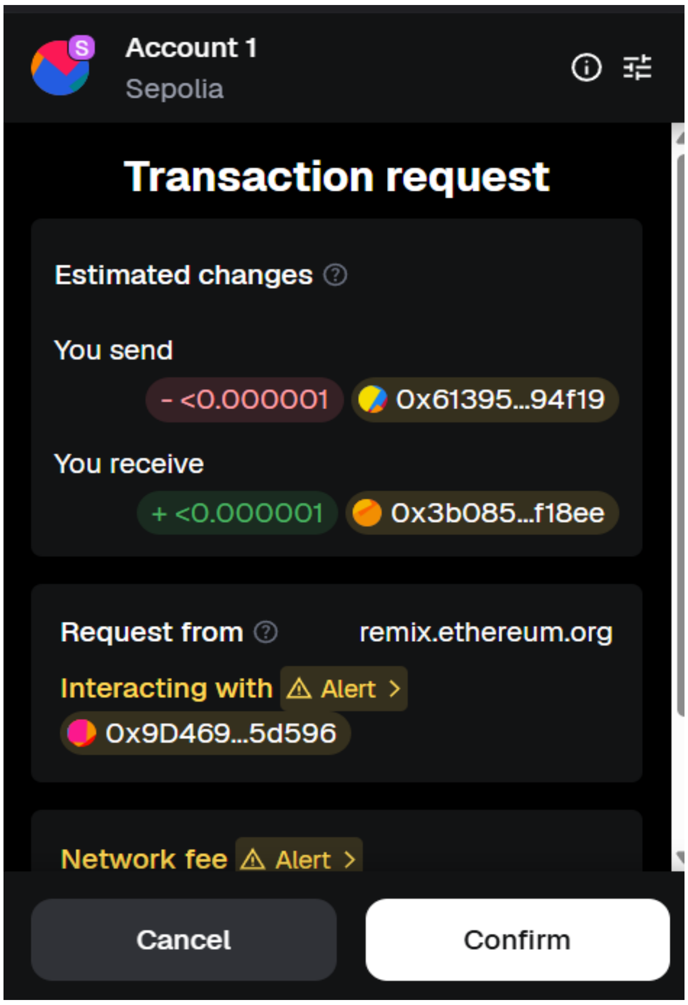
Estimated changes

You send
- <0.000001 0x61395...94f19
- <0.000001 0x3b085...f18ee

Request from remix.ethereum.org

Interacting with ⚠ Alert
0x9D469...5d596

Network fee
0.0002 SepoliaETH



Account 1
Sepolia

Transaction request

Estimated changes

You send
- <0.000001 0x61395...94f19

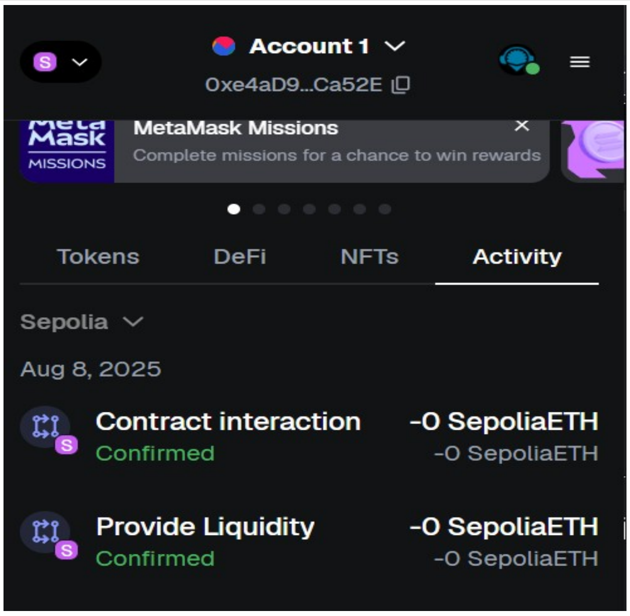
You receive
+ <0.000001 0x3b085...f18ee

Request from remix.ethereum.org

Interacting with ⚠ Alert
0x9D469...5d596

Network fee ⚠ Alert

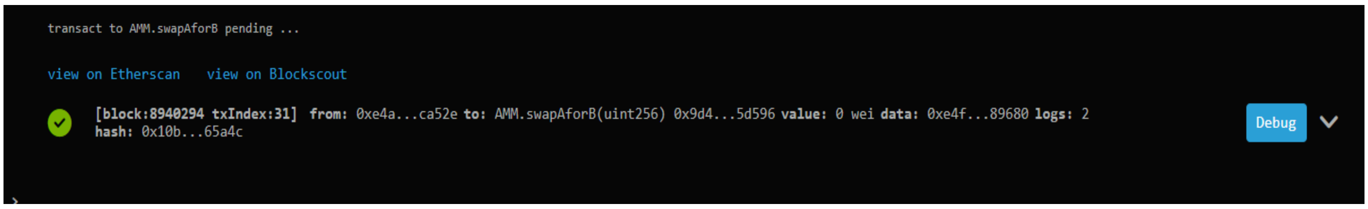
* Implementation Phase: Final Output (no error)



Aug 8, 2025

Contract interaction -0 SepoliaETH
Confirmed -0 SepoliaETH

Provide Liquidity -0 SepoliaETH
Confirmed -0 SepoliaETH



```
transact to AMM.swapAforB pending ...

view on Etherscan   view on Blockscout

[block:8940294 txIndex:31] from: 0xe4a...ca52e to: AMM.swapAforB(uint256) 0x9d4...5d596 value: 0 wei data: 0xe4f...89680 logs: 2
hash: 0x10b...65a4c
```

* Observations

1.The ERC-20 tokens (TokenA and TokenB) were successfully deployed and visible in MetaMask.
2.The AMM smart contract correctly handled liquidity addition for both tokens.
3.Swap transactions were executed successfully, and token balances updated as expected.
4.All transactions were confirmed on the Ethereum test network without errors.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

******As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.