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 : Build DeFi – AMM or Lending Prototype

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ALGORITHM:

```
1.Start
2.Create two tokens and name them as per your own
3.Deploy both the tokens
4.Note down the deployed contract addresses for both of the tokens
5.Open your wallet and import both the tokens
6.Now write the AMM (Automated Market Maker) smart contract code
7.Compile the AMM smart contract code
8.Deploy the AMM contract on the same network as your tokens
9.Note down the AMM contract address
10.Now from your wallet approve the AMM contract to spend a chosen number from tokenA. Do the same for tokenB
11.Call the AMM contract's addLiquidity function to deposit tokenA and tokenB into the pool
12.Verify that the liquidity has been added successfully by checking reserves
13.Now call the AMM contract's swap function to exchange token with one another
14.Confirm swap transaction and then check your wallet for updated balance
15.End
```

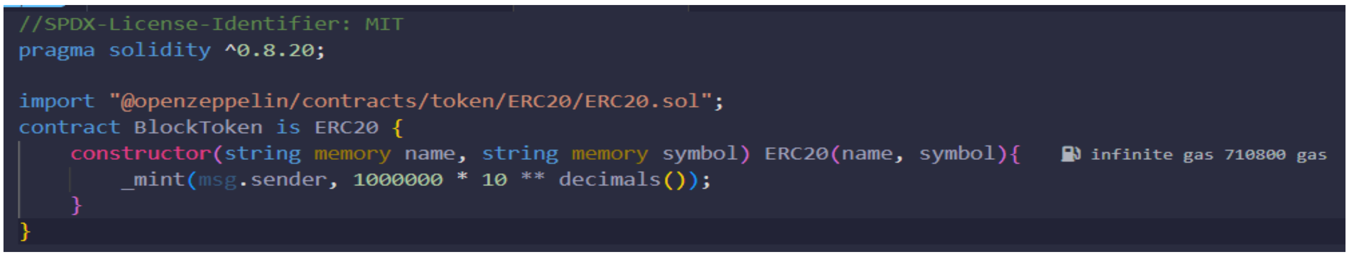
* Software used

1.Remix IDE
2.Metamask wallet
3.Sepolia test network
4.Etherscan testnet explorer
5.Brave browser

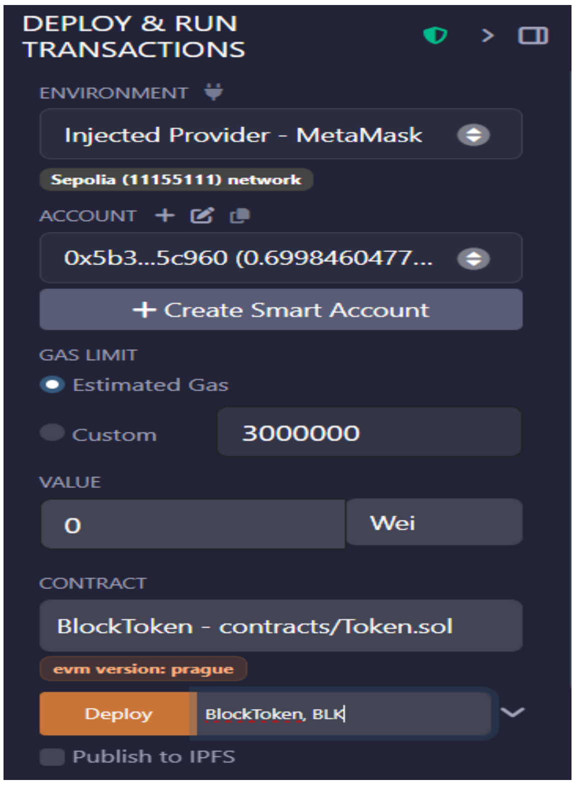
*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Testing Phase: Compilation of Code (error detection)

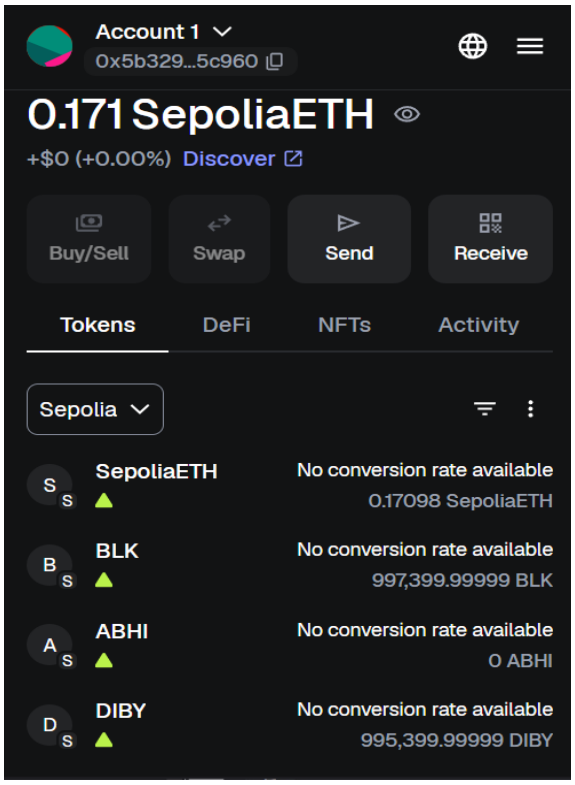
First create your two tokens using ERC20 i have already created two token one is BlockToken and another is DibyaToken and i already import them in my metamask wallet. This is the smart contract for creating your own token, after compiling the smart contract in deploy time we have to pass the string token name and symbol of our token (e.g-BlockToken, BLK) 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 BlockToken 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: 0x5b3...5c960 (0.6998460477...
+ Create Smart Account
GAS LIMIT: Estimated Gas
Custom: 3000000
VALUE: 0 Wei
CONTRACT: BlockToken - contracts/Token.sol
evm version: prague
Deploy: BlockToken, BLK
Publish to IPFS



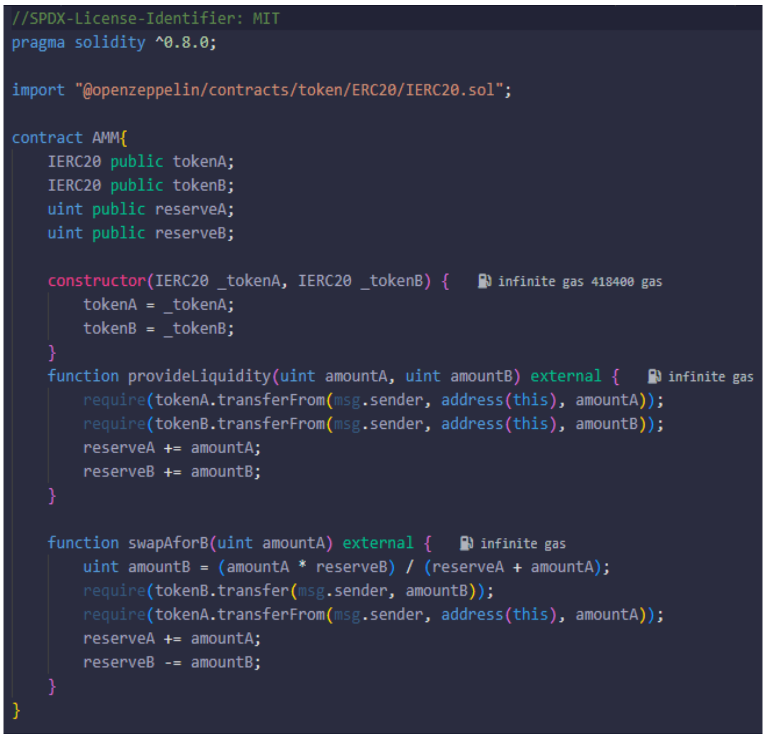
Account 1
0x5b329...5c960
0.171 SepoliaETH
+$0 (+0.00%) Discover
Buy/Sell · Swap · Send · Receive
Tokens · DeFi · NFTs · Activity
Sepolia

SepoliaETH	No conversion rate available	0.17098 SepoliaETH
BLK	No conversion rate available	997,399.99999 BLK
ABHI	No conversion rate available	0 ABHI
DIBY	No conversion rate available	995,399.99999 DIBY

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

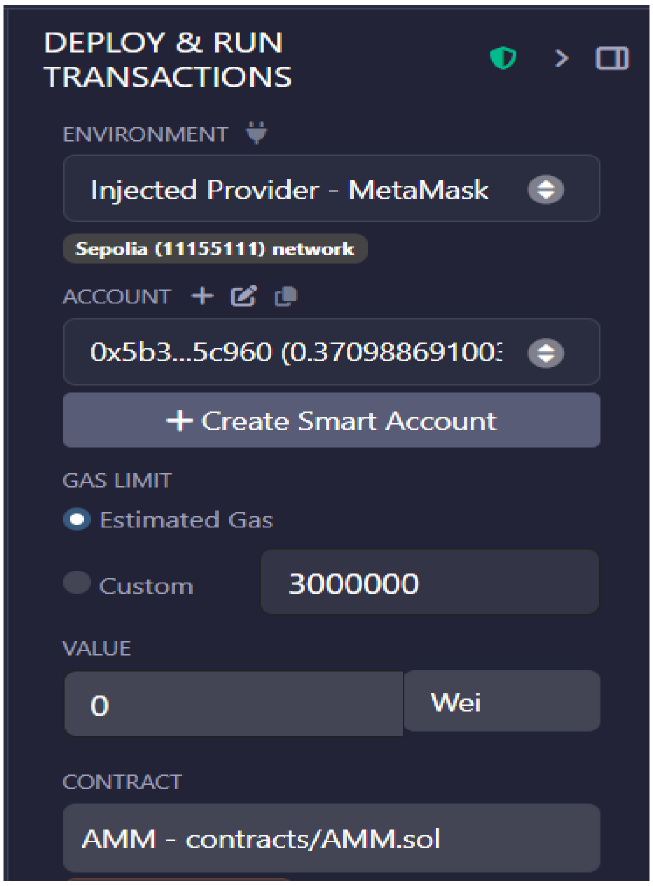
* Testing Phase: Compilation of Code (error detection)

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



```
//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) {    infinite gas 418400 gas
        tokenA = _tokenA;
        tokenB = _tokenB;
    }
    function provideLiquidity(uint amountA, uint amountB) external {    infinite gas
        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 {    infinite gas
        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 successful compilation we have to deploy the smart contract before deploying the smart contract first we have to choose the injector provider as metamask



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT
Injected Provider - MetaMask
Sepolia (11155111) network

ACCOUNT
0x5b3...5c960 (0.370988691003

+ Create Smart Account

GAS LIMIT
● Estimated Gas
○ Custom 3000000

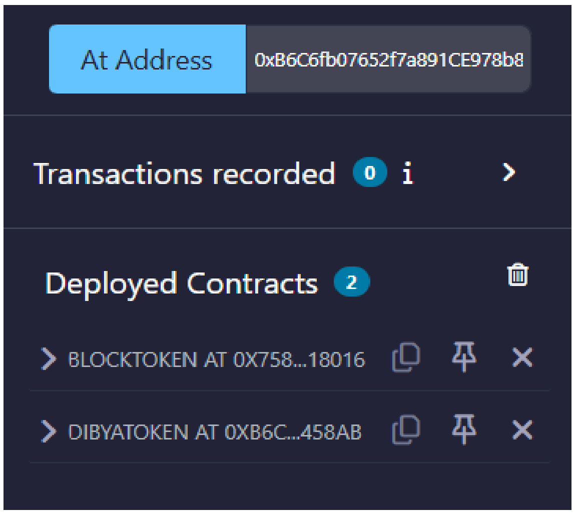
VALUE
0 Wei

CONTRACT
AMM - contracts/AMM.sol

*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 previously deployed ERC20 tokens in the deployed section


At Address 0xB6C6fb07652f7a891CE978b8

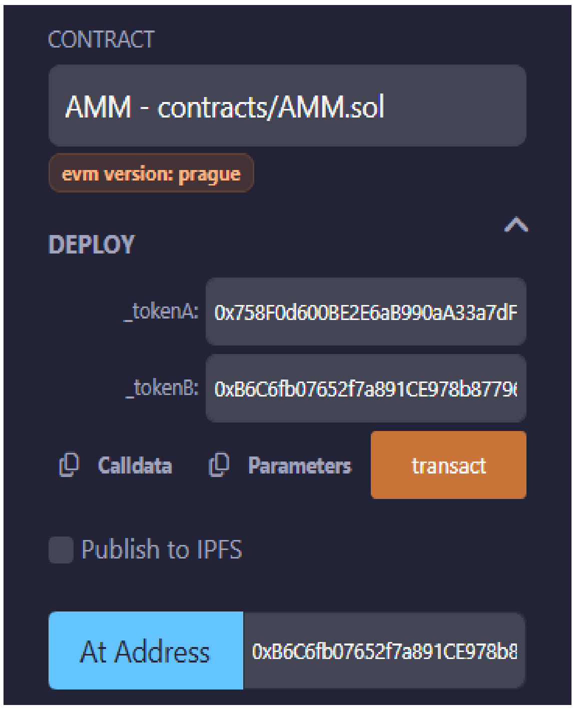
Transactions recorded 0

Deployed Contracts 2

BLOCKTOKEN AT 0X758...18016

DIBYATOKEN AT 0XB6C...458AB

Now deploy AMM smart contract by giving the contract address of both the ERC20 tokens.


CONTRACT

AMM - contracts/AMM.sol

evm version: prague

DEPLOY

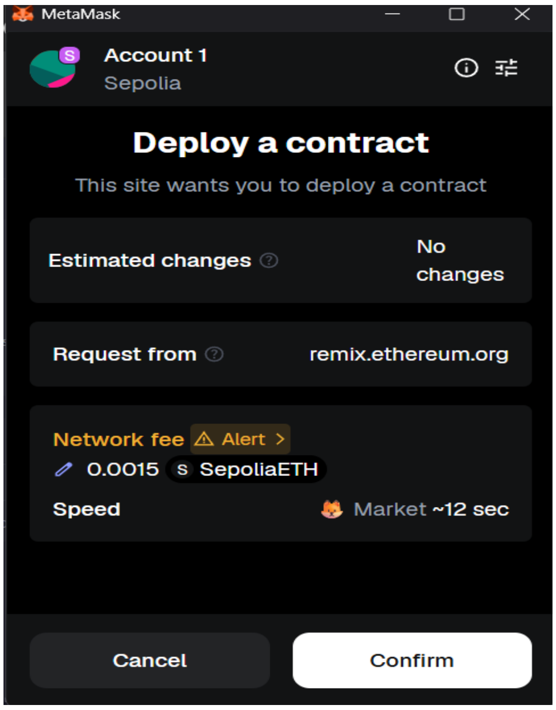
_tokenA: 0x758F0d600BE2E6aB990aA33a7dF

_tokenB: 0xB6C6fb07652f7a891CE978b8779

Calldata Parameters transact

Publish to IPFS

At Address 0xB6C6fb07652f7a891CE978b8


MetaMask

Account 1
Sepolia

Deploy a contract

This site wants you to deploy a contract

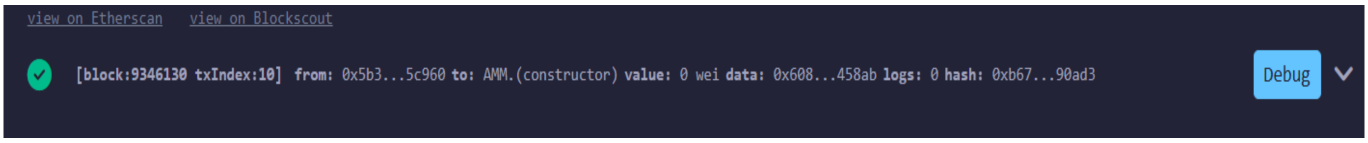
Estimated changes: No changes

Request from: remix.ethereum.org

Network fee ⚠ Alert
0.0015 S SepoliaETH

Speed: Market ~12 sec

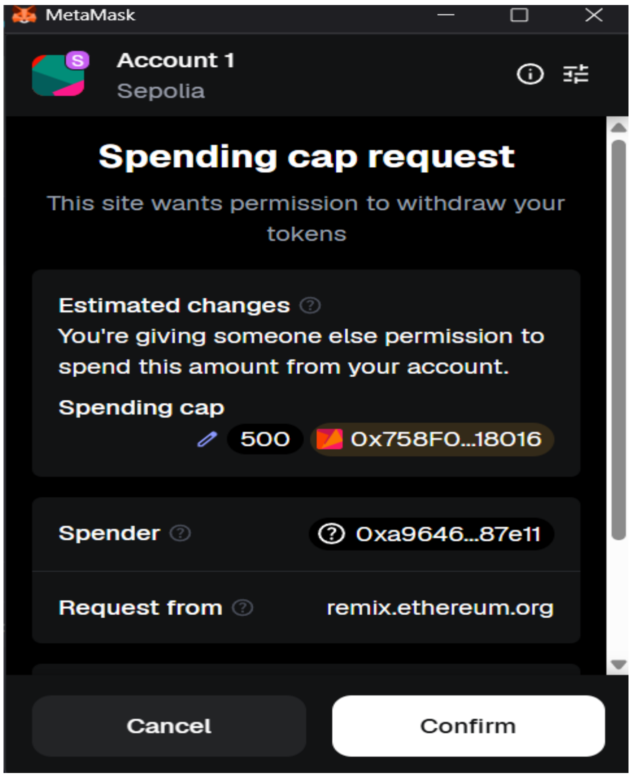
Cancel Confirm


```
view on Etherscan   view on Blockscout

[block:9346130 txIndex:10] from: 0x5b3...5c960 to: AMM.(constructor) value: 0 wei data: 0x608...458ab logs: 0 hash: 0xb67...90ad3
```

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

* Testing Phase: Compilation of Code (error detection)

Now copy the AMM contract address and paste it in the approve function of token1 and give some uint value to be transferred and do the same in transfer function



MetaMask

Account 1
Sepolia

Spending cap request

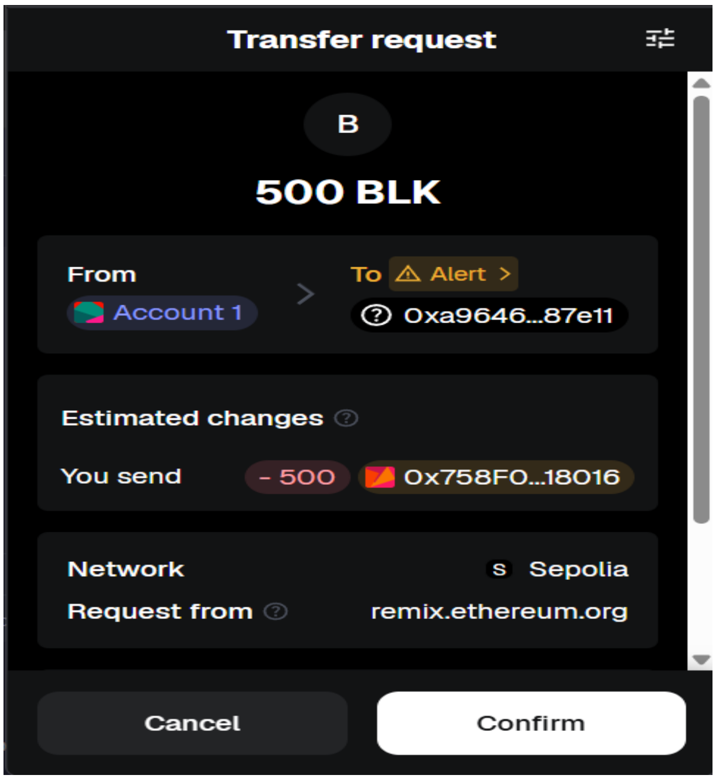
This site wants permission to withdraw your tokens

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

Spending cap: 500 0x758F0...18016

Spender: 0xa9646...87e11

Request from: remix.ethereum.org



Transfer request

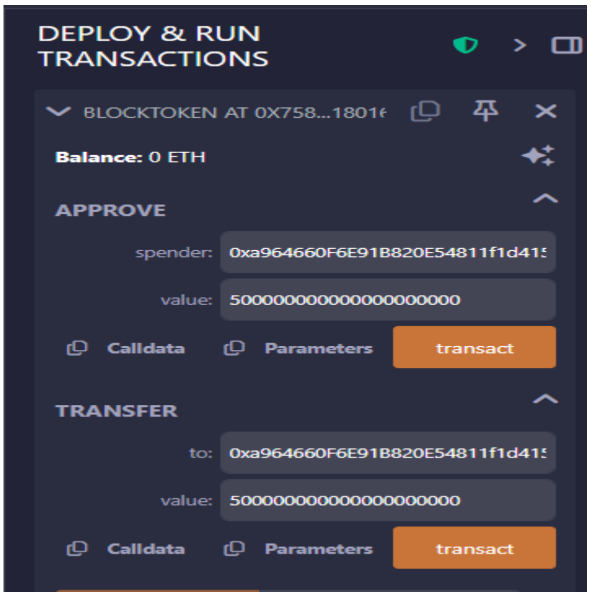
B

500 BLK

From: Account 1
To ⚠ Alert: 0xa9646...87e11

Estimated changes
You send: - 500 0x758F0...18016

Network: Sepolia
Request from: remix.ethereum.org



DEPLOY & RUN TRANSACTIONS

BLOCKTOKEN AT 0X758...18016

Balance: 0 ETH

APPROVE

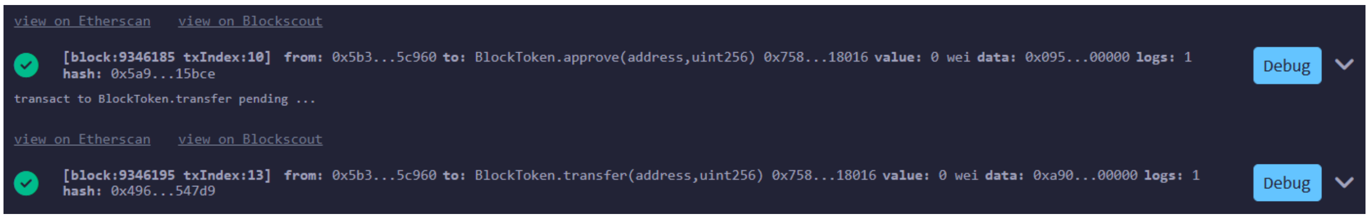
spender: 0xa964660F6E91B820E54811f1d41

value: 500000000000000000000

TRANSFER

to: 0xa964660F6E91B820E54811f1d41

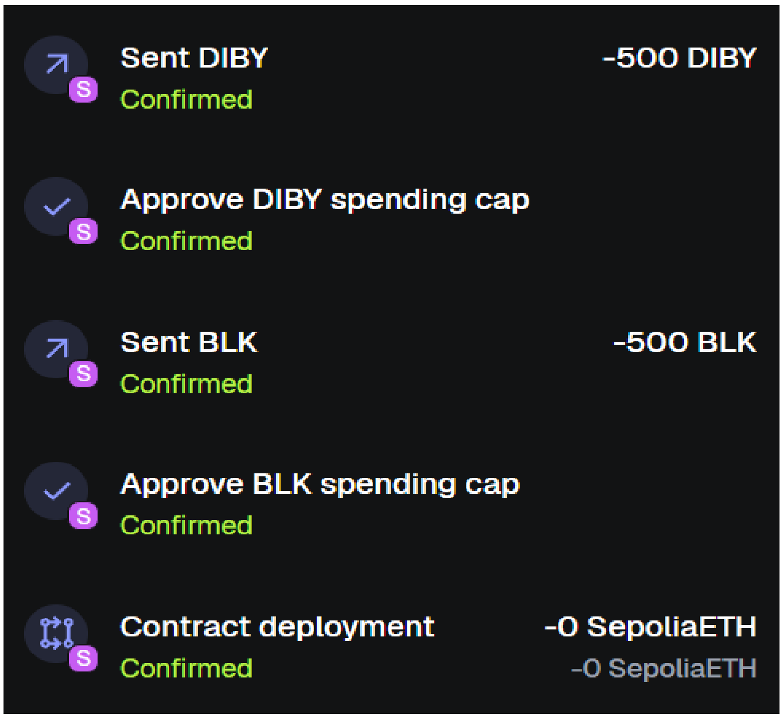
value: 500000000000000000000



```
view on Etherscan    view on Blockscout

[block:9346185 txIndex:10]  from: 0x5b3...5c960 to: BlockToken.approve(address,uint256) 0x758...18016 value: 0 wei data: 0x095...00000 logs: 1
hash: 0x5a9...15bce
transact to BlockToken.transfer pending ...

view on Etherscan    view on Blockscout

[block:9346195 txIndex:13]  from: 0x5b3...5c960 to: BlockToken.transfer(address,uint256) 0x758...18016 value: 0 wei data: 0xa90...00000 logs: 1
hash: 0x496...547d9
```

Now follow the same steps for token2

* Testing Phase: Compilation of Code (error detection)


Sent DIBY -500 DIBY
Confirmed

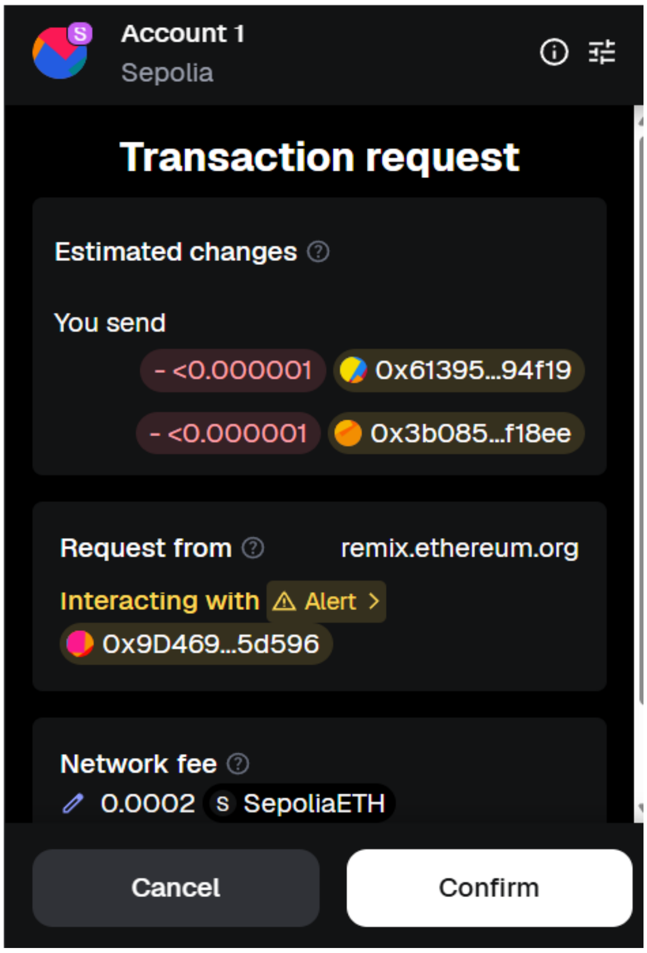
Approve DIBY spending cap
Confirmed

Sent BLK -500 BLK
Confirmed

Approve BLK spending cap
Confirmed

Contract deployment -0 SepoliaETH
Confirmed -0 SepoliaETH

Now after giving access to the token its time to check the provide liquidity to check liquidity given to amountA and amountB


Account 1
Sepolia

Transaction request

Estimated changes

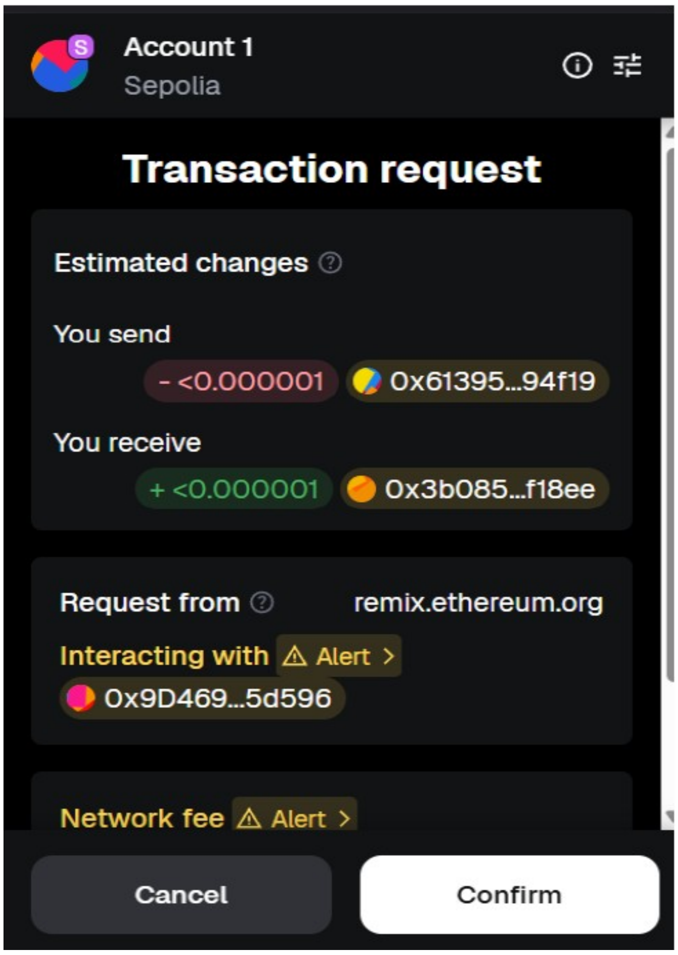
You send
- <0.000001 0x61395...94f19
- <0.000001 0x3b085...f18ee

Request from remix.ethereum.org

Interacting with ⚠ Alert >
0x9D469...5d596

Network fee
0.0002 SepoliaETH

Cancel Confirm


Account 1
Sepolia

Transaction request

Estimated changes

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

You receive
+ <0.000001 0x3b085...f18ee

Request from remix.ethereum.org

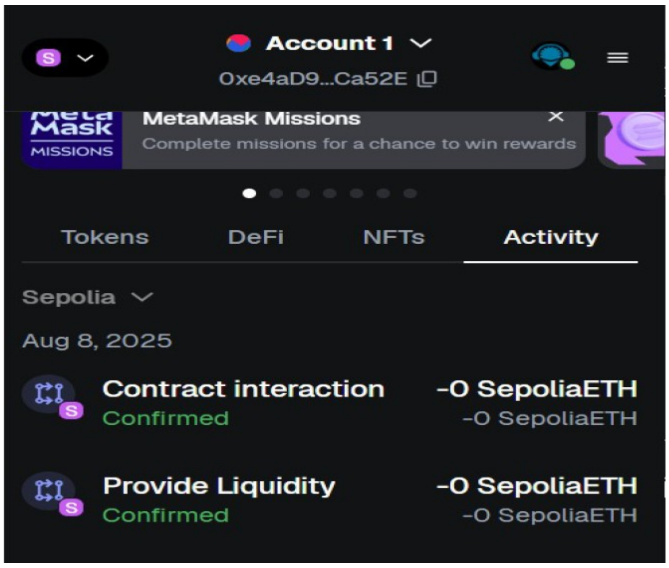
Interacting with ⚠ Alert >
0x9D469...5d596

Network fee ⚠ Alert >

Cancel Confirm

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

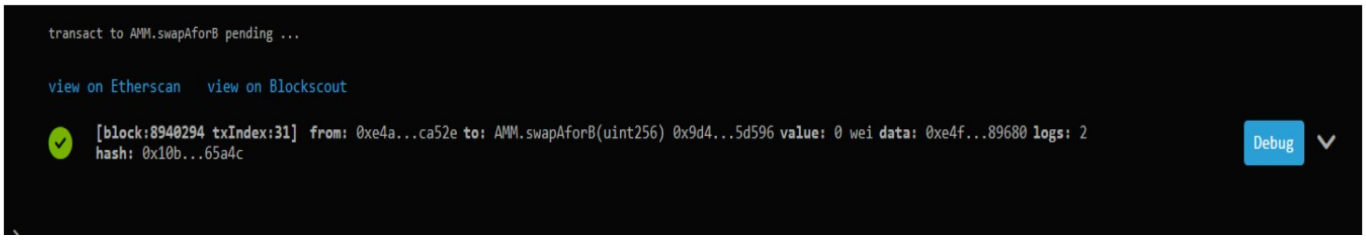
* Implementation Phase: Final Output (no error)



Aug 8, 2025

Contract interaction — Confirmed — -0 SepoliaETH / -0 SepoliaETH

Provide Liquidity — Confirmed — -0 SepoliaETH / -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 DeFi lab helps understand how decentralized finance protocols like AMMs or lending platforms work through smart contracts and liquidity management.

2. It provides hands-on experience in building, testing, and deploying Solidity-based financial systems on blockchain.

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.