



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 Experiment :

* 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

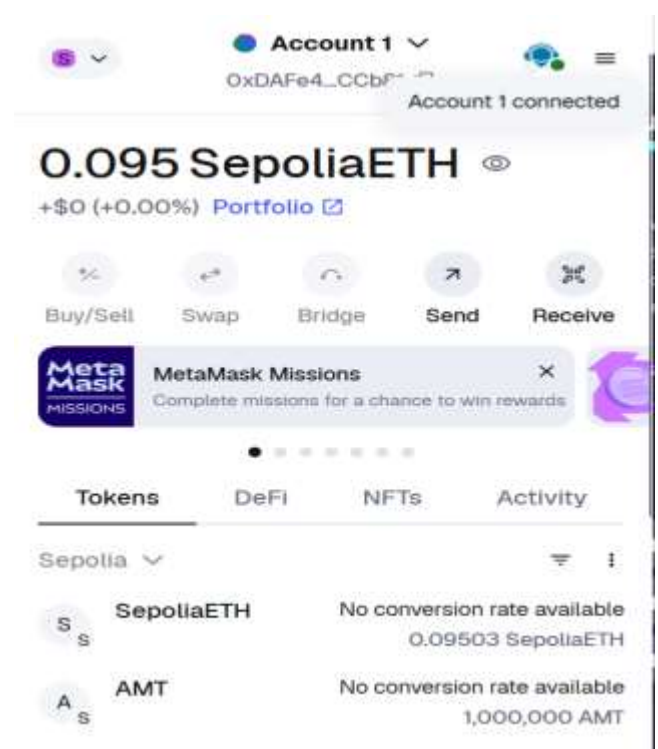
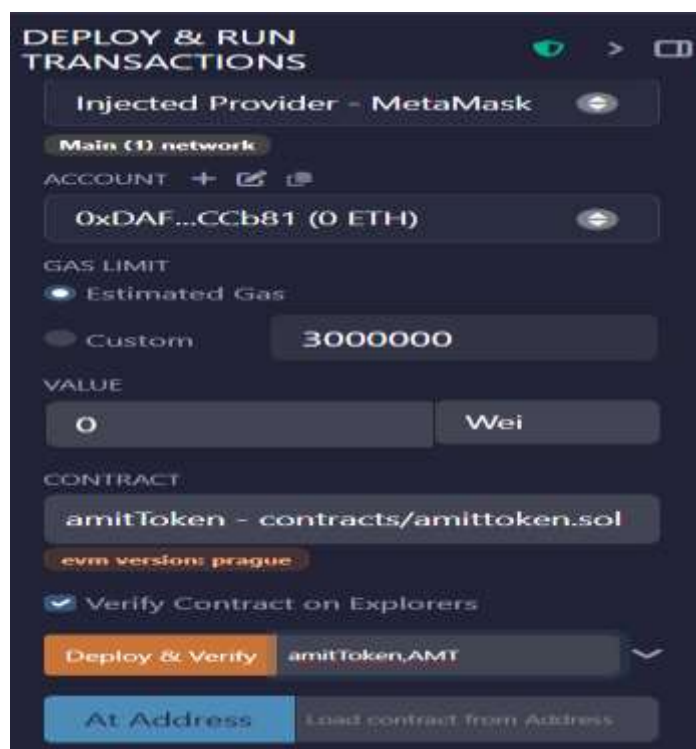
* Testing Phase: Compilation of Code (error detection)

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.



```

1 // SPDX-License-Identifier: MIT
2 // Compatible with OpenZeppelin Contracts ^5.0.0
3 pragma solidity ^0.8.27;
4
5 import {ERC20} from "@openzeppelin/contracts@5.3.0/token/ERC20/ERC20.sol";
6
7 contract amittoken is ERC20 {
8
9     constructor(string memory name, string memory symbol) {
10         ERC20(name, symbol) {
11             _mint(msg.sender, 1000000 * 10 ** decimals());
12         }
13     }
14 }
  
```



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

* Testing Phase: Compilation of Code (error detection)

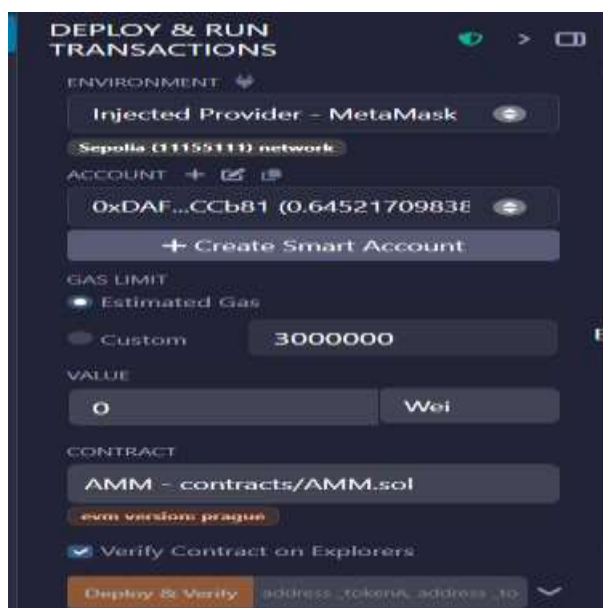
The smart contract for AMM is including functions like providesolidity and swapforAandB .

```

1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3  import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
4  contract AMM {
5      IERC20 public tokenA;
6      IERC20 public tokenB;
7      uint public reserveA;
8      uint public reserveB;
9      constructor(IERC20 _tokenA, IERC20 _tokenB) {
10         tokenA = _tokenA;
11         tokenB = _tokenB;
12     }
13     function provideLiquidity(uint amountA, uint amountB) external {
14         require(tokenA.transferFrom(msg.sender, address(this), amountA));
15         require(tokenB.transferFrom(msg.sender, address(this), amountB));
16         reserveA += amountA;
17         reserveB += amountB;
18     }
19     function swapAforB(uint amountA) external {
20         uint amountB = (amountA * reserveB) / (reserveA + amountA);
21         require(tokenB.transfer(msg.sender, amountB));
22         require(tokenA.transferFrom(msg.sender, address(this), amountA));
23         reserveA += amountA;
24         reserveB -= amountB;
25     }
26 }
27

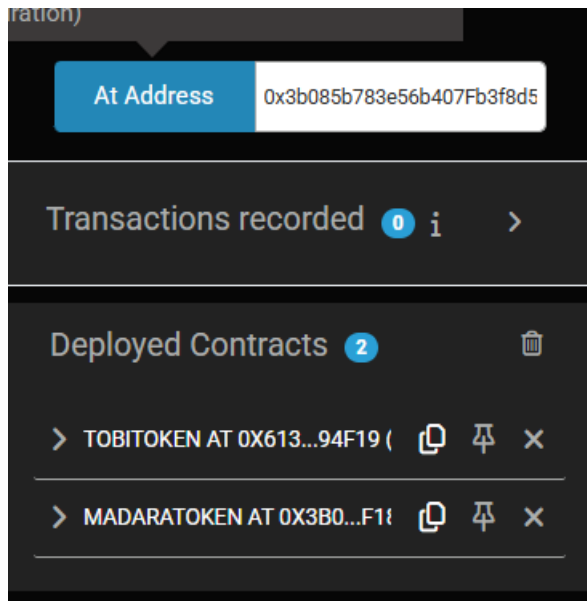
```

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

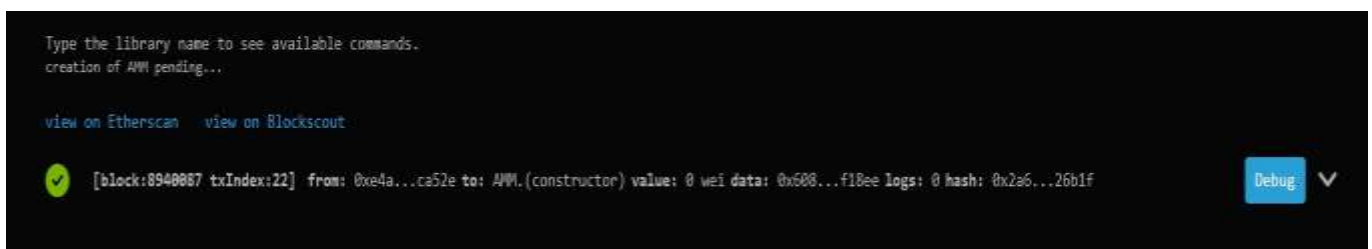
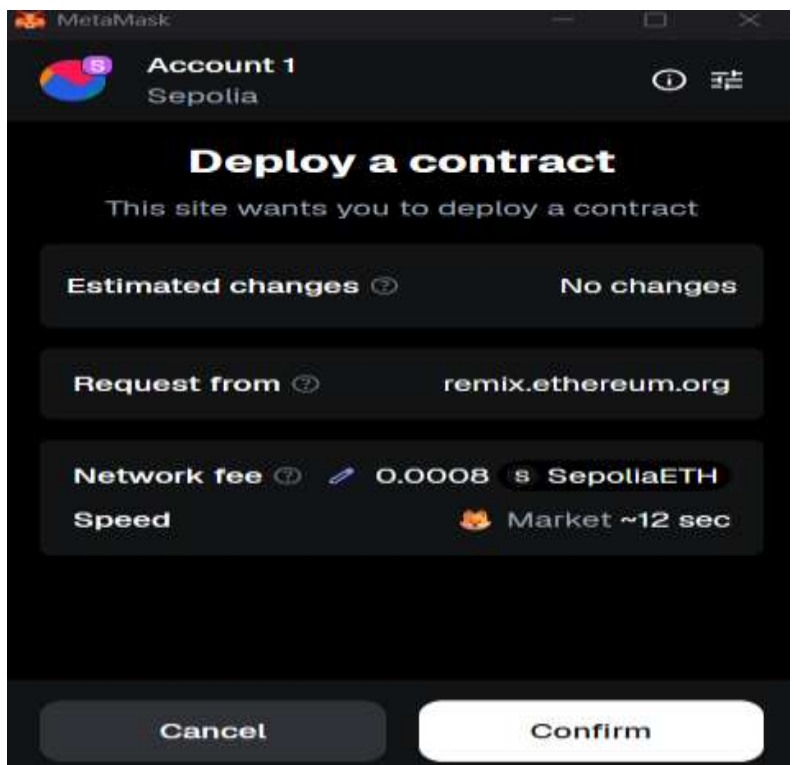
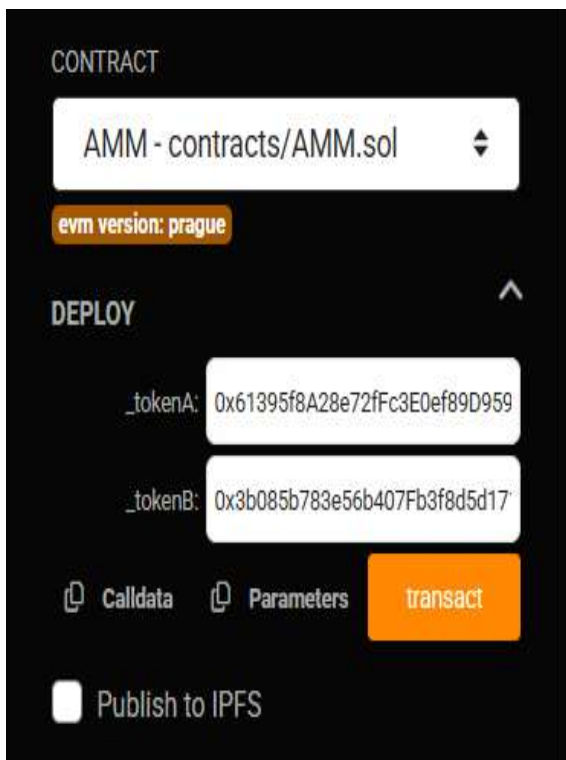


* Testing Phase: Compilation of Code (error detection)

Now add two previous deployed ERC20 tokens in the deployed section

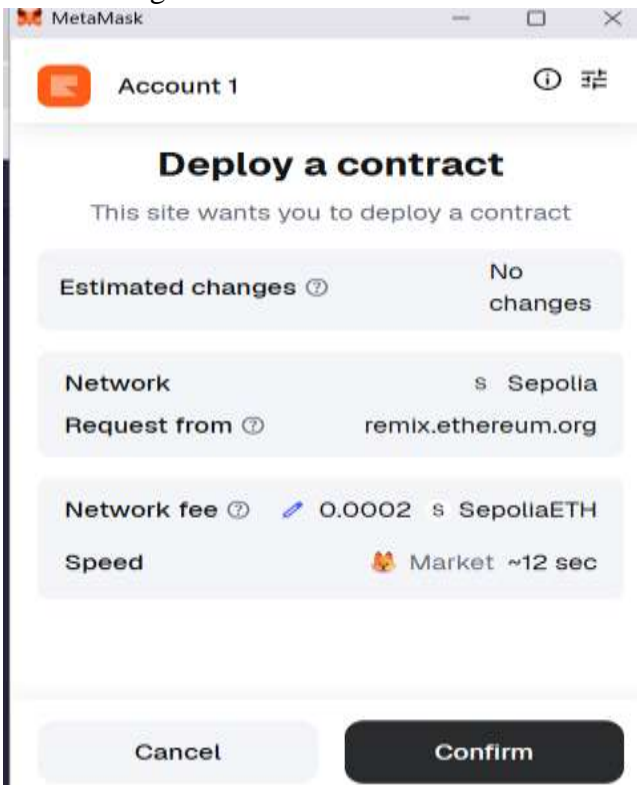


Now Deploy the AMM smart contract to deploy the smart contract we have to give the contract address of two tokens



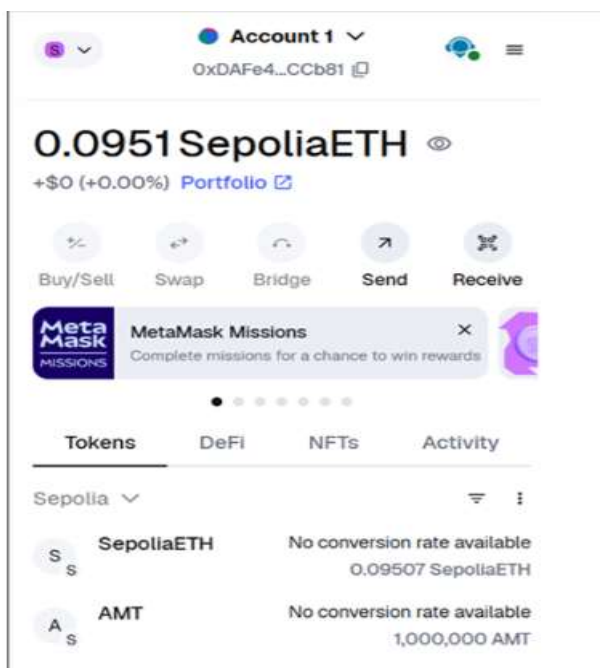
* Testing Phase: Compilation of Code (error detection)

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.

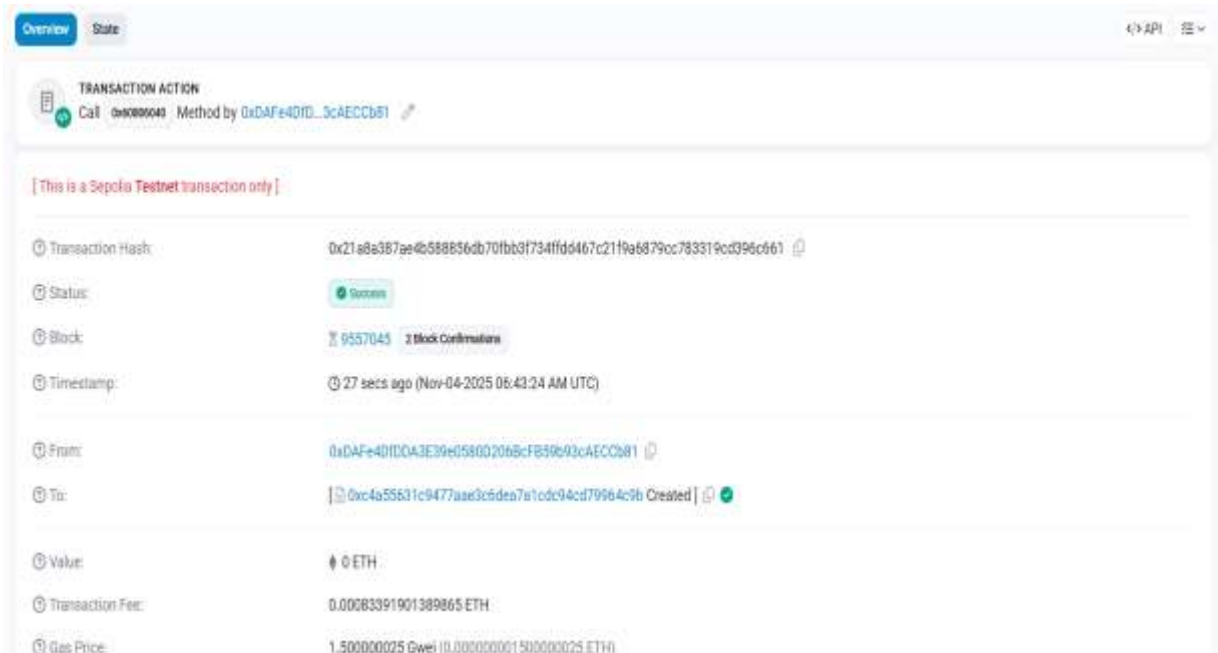


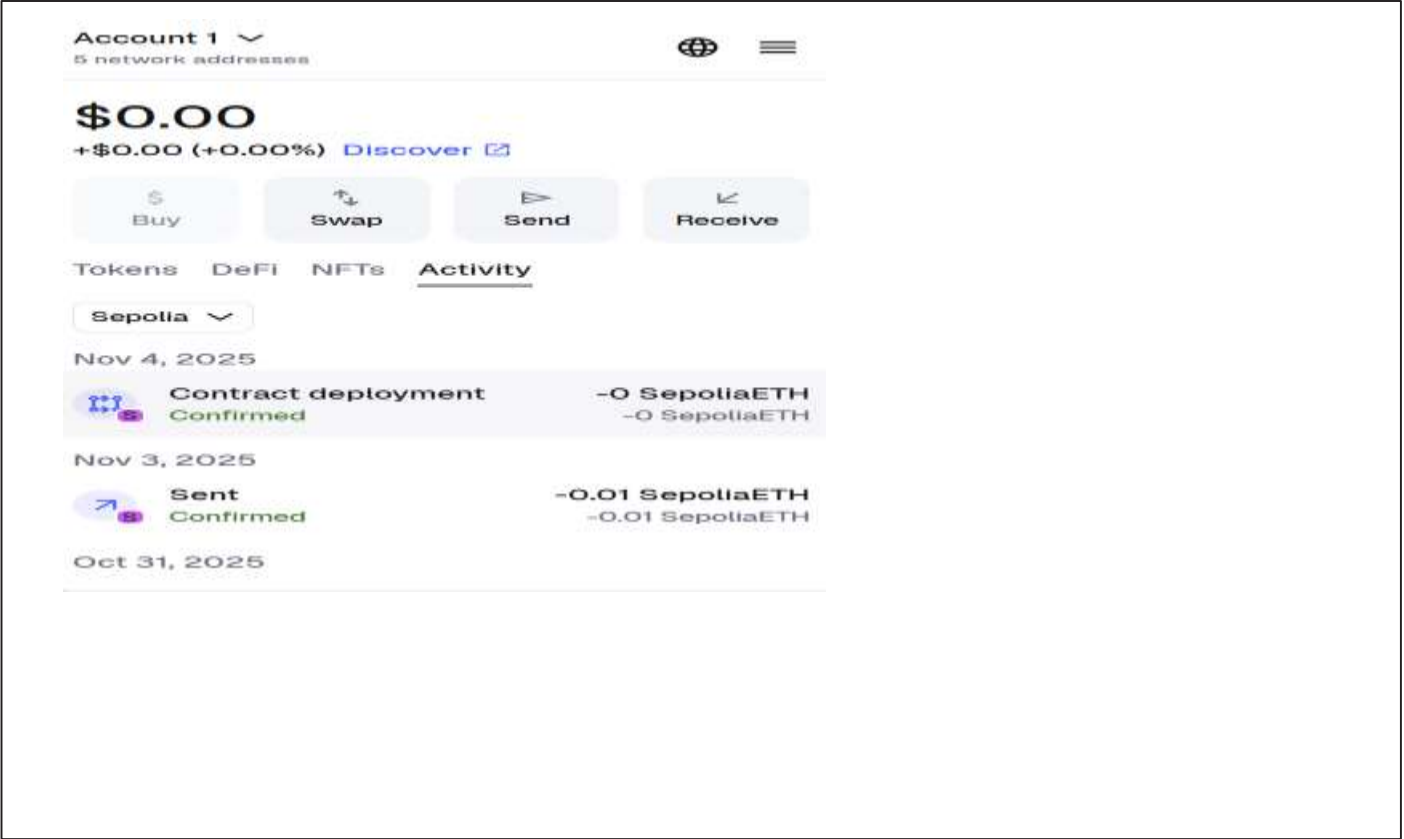
Now do the same steps for the another token MAD.

* Testing Phase: Compilation of Code (error detection)



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





* 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		