



Centurion
UNIVERSITY
Mysore City
Karnataka, India

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

Page No.....

*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 etherscan 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.

```

    ✓ Compiled   🔍 🔍 Home amittoken.sol ✘
1 // SPDX-License-Identifier: MIT
2 // Compatible with OpenZeppelin Contracts >v5.0.0
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) infinite gas 710000 gas
10    ERC20(name, symbol) {
11        _mint(msg.sender, 1000000 * 10 ** decimals());
12    }
13}

```

DEPLOY & RUN TRANSACTIONS

Injected Provider - MetaMask

Main (1) network

ACCOUNT: OxD...CCb81 (0 ETH)

GAS LIMIT: Estimated Gas: 3000000

VALUE: 0 Wei

CONTRACT: amitToken - contracts/amittoken.sol

evm version: prague

Verify Contract on Explorers

Deploy & Verify: amitToken, AMT

At Address: Load contract from Address

0.095 SepoliaETH

+\$0 (+0.00%) Portfolio

Buy/Sell Swap Bridge Send Receive

MetaMask MISSIONS Complete missions for a chance to win rewards

Tokens DeFi NFTs Activity

Sepolia

S S	SepoliaETH	No conversion rate available 0.09503 SepoliaETH
A S	AMT	No conversion rate available 1,000,000 AMT

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)

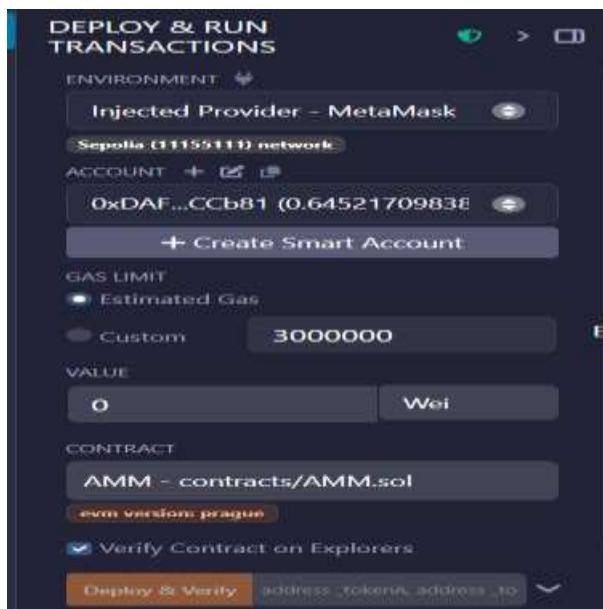
The smart contract for AMM is including functions like providesolidity and swapforAandB .

```

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 swapForB(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

The screenshot shows the MetaMask wallet interface. At the top, there is a blue button labeled "At Address" with the address "0x3b085b783e56b407Fb3f8d5". Below this, a section titled "Transactions recorded" shows 0 transactions. Under "Deployed Contracts", there are two entries: "TOBITOKEN AT 0X613...94F19" and "MADARATOKEN AT 0X3B0...F18". Each entry has icons for "View", "Edit", and "Delete".

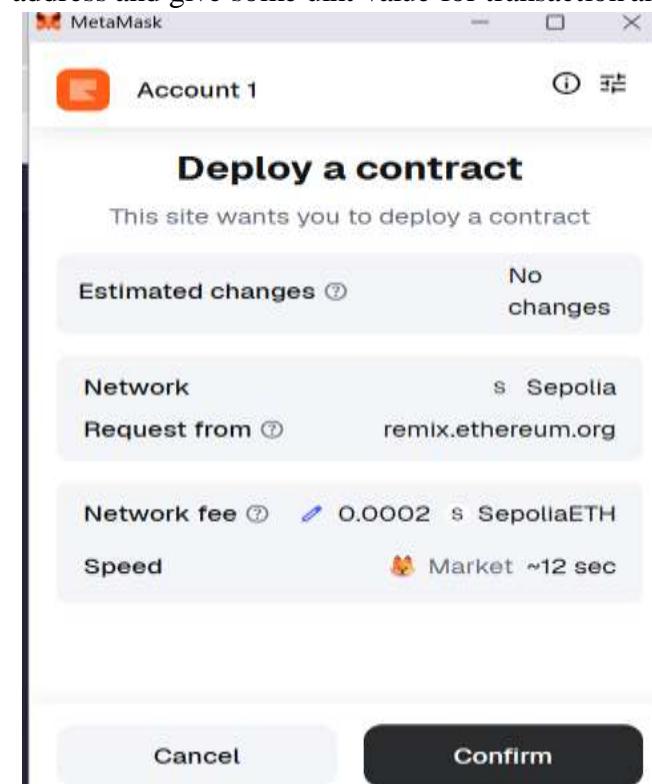
Now Deploy the AMM smart contract to deploy the smart contract we have to give the contract address of two tokens

The screenshot shows the Remix IDE interface. On the left, under "CONTRACT", it says "AMM - contracts/AMM.sol" and "evm version: prague". Under "DEPLOY", it shows two input fields: "_tokenA: 0x61395f8A28e72ffC3E0ef89D959" and "_tokenB: 0x3b085b783e56b407Fb3f8d5d17". Below these are buttons for "Calldata", "Parameters", and an orange "transact" button. On the right, a modal window titled "Deploy a contract" is open. It displays the message "This site wants you to deploy a contract". It shows "Estimated changes" (No changes), "Request from" (remix.ethereum.org), "Network fee" (0.0008 SepoliaETH), "Speed" (Market ~12 sec), and two large buttons at the bottom: "Cancel" and "Confirm".

The screenshot shows the Etherscan interface. It displays the deployment information for the AMM contract. It says "creation of AMM pending..." and provides links to "view on Etherscan" and "view on Blockscout". At the bottom, it shows a green checkmark icon followed by the transaction details: "[block:8940867 txIndex:22] from: 0xe4a...ca52e to: AMM.(constructor) value: 0 wei data: 0x500...f18ee logs: 0 hash: 0x2a6...26b1f". To the right of the transaction details are "Debug" and "View" buttons.

* 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 smae steps for the another token MAD.

* Testing Phase: Compilation of Code (error detection)

The screenshot shows the MetaMask interface for Account 1 (Address: 0xDAFe4...CCb81). The balance is 0.0951 SepoliaETH. Below the balance, there are buttons for Buy/Sell, Swap, Bridge, Send, and Receive. An overlay window titled "MetaMask Missions" is displayed, stating "Complete missions for a chance to win rewards". The main content area shows tokens: SepoliaETH (0.09507 SepoliaETH) and AMT (1,000,000 AMT).

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

The screenshot shows the Etherscan transaction details for a Sepolia Testnet transaction. The transaction hash is 0x21a8e387ae4b588856db70fb3f734ffd467c21f9a6879cc783319cd396c661. It was successful (Status: Success) in Block 9557045 with 2 block confirmations. The timestamp is 27 secs ago (Nov-04-2025 06:43:24 UTC). The transaction originated from 0xDAFe40D043E39e058002068cFB50e03cAECCb81 and went to 0xc4a55631c9477aae3c6de7a1cdc94cd79964c9b. The value transferred was 0 ETH, and the transaction fee was 0.00083391901389865 ETH (1.500000025 Gwei). The gas price was 0.0000001500000025 ETH.

* Implementation Phase: Final Output (no error)

Applied and Action Learning

The screenshot shows the MetaMask wallet interface. At the top, it displays "Account 1" with 5 network addresses. The balance is shown as \$0.00 with a +\$0.00 (+0.00%) change. Below the balance are four buttons: Buy, Swap, Send, and Receive. Underneath these buttons are tabs for Tokens, DeFi, NFTs, and Activity, with Activity being the selected tab. A dropdown menu shows "Sepolia". The activity log lists three transactions:

- Nov 4, 2025: Contract deployment (Confirmed) -0 SepoliaETH (-0 SepoliaETH)
- Nov 3, 2025: Sent (Confirmed) -0.01 SepoliaETH (-0.01 SepoliaETH)
- Oct 31, 2025: (No transaction listed)

* 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. :

Page No.....

Signature of the Faculty:

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