



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 : Build DeFi – AMM or Lending Prototype

Objective/Aim:

Perform basic operations such as swapping tokens (AMM) or depositing/borrowing assets (Lending), and record transactions and observations.

Apparatus/Software Used:

1. MetaMask Wallet
2. Brave Web Browser
3. Uniswap Testnet or Aave Testnet Website
4. Ethereum Sepolia Testnet Network

Theory/Concept:

Automated Market Maker (AMM)

AMM platforms like Uniswap allow users to **swap tokens** using **liquidity pools** instead of traditional order books.

- Liquidity providers deposit token pairs
- Users swap tokens directly against the pool
- Pools adjust prices using mathematical formulas such as $x \times y = k$

Lending Protocol (Aave Model)

A lending protocol enables:

- Depositing assets → earn interest
 - Borrowing assets → using collateral
 - Interest rates determined by supply–demand
- On testnets, all functions work the same but with test tokens.

Procedure:

1. Open MetaMask → switch to **Sepolia Testnet**.
2. Visit aave.com, click **Enter App**.
3. Connect Wallet → MetaMask.
4. Enable **Testnet Mode** and choose **Ethereum Sepolia**.
5. Select **Deposit** and choose an asset (e.g., ETH).
6. Enter deposit amount → confirm.
7. Next, select **Borrow** to borrow a test asset (e.g., DAI).
8. Review gas fees and loan details → confirm.
9. Observe dashboard for health factor, interest rate, and collateral value.

Solidity contract

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6 contract tobiToken is ERC20 {
7     constructor(string memory name, string memory symbol) ERC20(name, symbol) {
8         _mint(msg.sender, 1000000 * 10 ** decimals());
9     }
10 }
11

```

The screenshot shows the REMIX IDE interface. On the left, the **DEPLOY & RUN TRANSACTIONS** panel is active, displaying the environment as **Sepolia (11155111) network**, account **0x234...E3f45 (0.152083586272)**, and a gas limit of **3000000**. On the right, the **CONTRACT** panel shows the deployed contract **MyToken - MyToken.sol** and the EVM version **prague**. The EVM panel lists four tokens with their names and conversion rates:

- @g: No conversion rate available, 999,400 @g
- ##: No conversion rate available, 999,000 ##
- EEC: No conversion rate available, 100.00B EEC
- EXC: No conversion rate available, 100.00M EXC

```

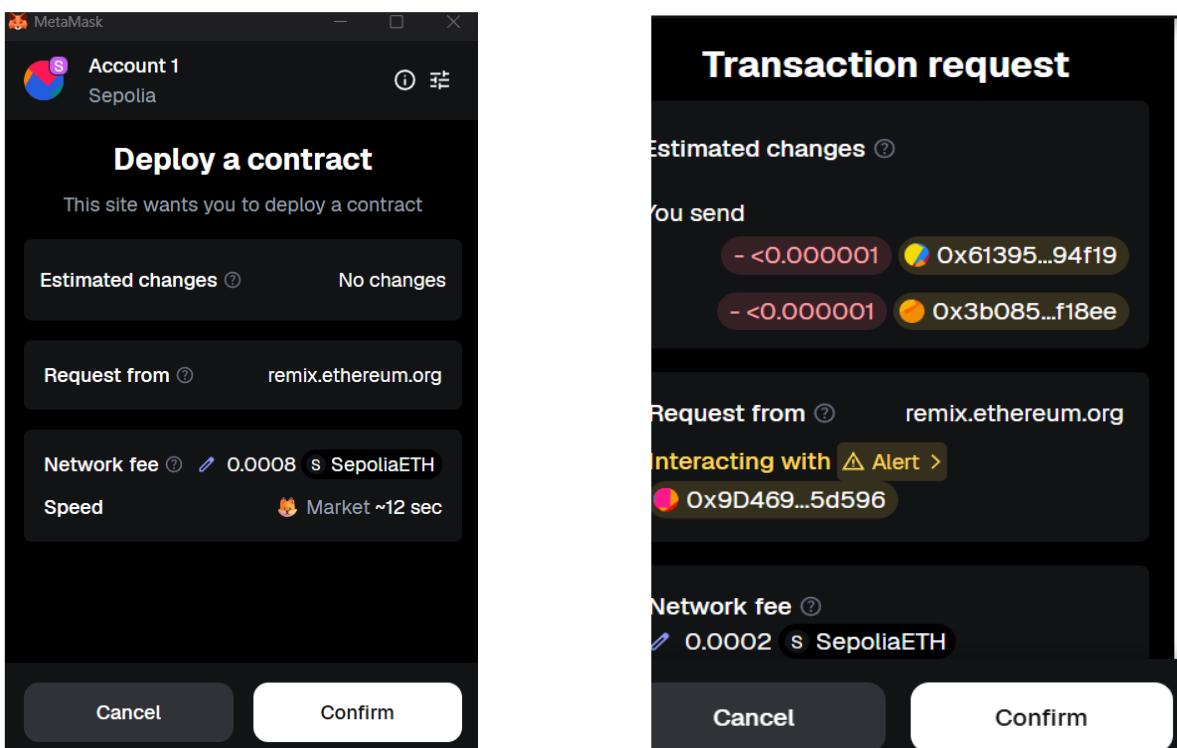
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
4 contract AMM {
5     IERC20 public tokenA;
6     IERC20 public tokenB;
7     uint public reserveA;
8     uint public reserveB;
9     constructor(IERC20 _tokenA, IERC20 _tokenB) {
10         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

```

□

This screenshot shows a dashboard for managing smart contracts. On the left, there's a sidebar with a blue header labeled "At Address" containing the address "0x3b085b783e56b407Fb3f8d5". Below this, a section titled "Transactions recorded" shows 0 transactions with an info icon and a right-pointing arrow. At the bottom, a section titled "Deployed Contracts" shows 2 contracts: "TOBITOKEN AT 0X613...94F19" and "MADARATOKEN AT 0X3B0...F1". Each contract entry has a delete icon.

This screenshot shows the deployment interface for the "AMM - contracts/AMM.sol" contract. It includes a dropdown for "evm version: prague" and a "DEPLOY" button. The deployment parameters are set to "_tokenA: 0x61395f8A28e72fFc3E0ef89D959" and "_tokenB: 0x3b085b783e56b407Fb3f8d5d17". Below the deployment buttons are "Calldata" and "Parameters" links, and an orange "transact" button. At the bottom, there's a "Publish to IPFS" button.



Observation

Action Performed	Result	Notes
Wallet Connected	Successful	MetaMask connected to Sepolia
AMM Swap / Lending Action	Completed	Transaction confirmed on testnet
Gas Fee Visibility	Yes	Gas fees and slippage clearly shown
Token Transfer/Lending	Successful	Test ETH swapped/deposited/borrowed

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: