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 : Read the Chain – Web3.js Basics

Objective/Aim:

To understand the fundamentals of **Web3.js**, a JavaScript library used to interact with the Ethereum blockchain, and to perform basic blockchain operations such as connecting to the network, fetching accounts, and reading smart contract data.

Apparatus/Software Used:

- Node.js
- Web3.js library
- Ethereum test network (e.g., Ganache / Sepolia / Goerli)
- MetaMask (optional, for account management)
- VS Code or any code editor

Theory/Concept:

Web3.js is a JavaScript library that allows interaction with Ethereum blockchain nodes using RPC (Remote Procedure Calls).

- **Blockchain** is a decentralized ledger consisting of blocks.
- Each block has a block number, timestamp, transactions, and hash.
- Using **Web3.js**, developers can:
  - Connect to an Ethereum node (via HTTP or WebSocket).
  - Read chain data such as current block number, gas price, or balances.
  - Interact with smart contracts.

In this experiment, we focus on **reading the chain**, i.e., fetching basic blockchain information.

Procedure:

Step 1: Write Token Contract (in Remix IDE)
Step 2: Open Remix IDE → Select compiler **0.8.x** → Compile MyToken.sol.
Step 3: **Deploy Token**

- In Remix → "Deploy & Run Transactions" tab.
- Select **Injected Web3** (MetaMask connected to local testnet).

Step 4: **Verify on Avee.net**

- Copy deployed contract address.
- Go to **Avee.net** → Add Token → Paste contract address.
- Token appears in the dashboard.

Step 5: **Interact with Token**

- Check total supply.
- Transfer tokens between MetaMask accounts.
- Use Avee.net to approve and transfer tokens on behalf of another account.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.30;

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

contract MyToken2 is ERC20 {
    constructor(string memory name, string memory symbol)
        ERC20(name, symbol)
    {
        _mint(msg.sender, 1000000 * 10 ** decimals());
    }
}
```

Fig 1: smart contract

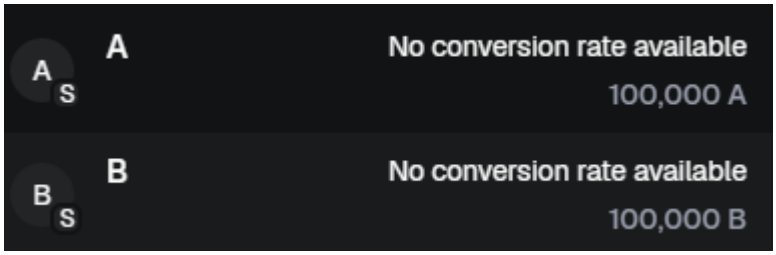


Fig2 : Tokens

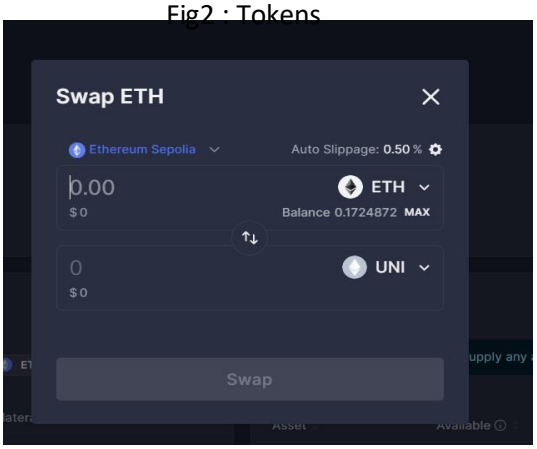
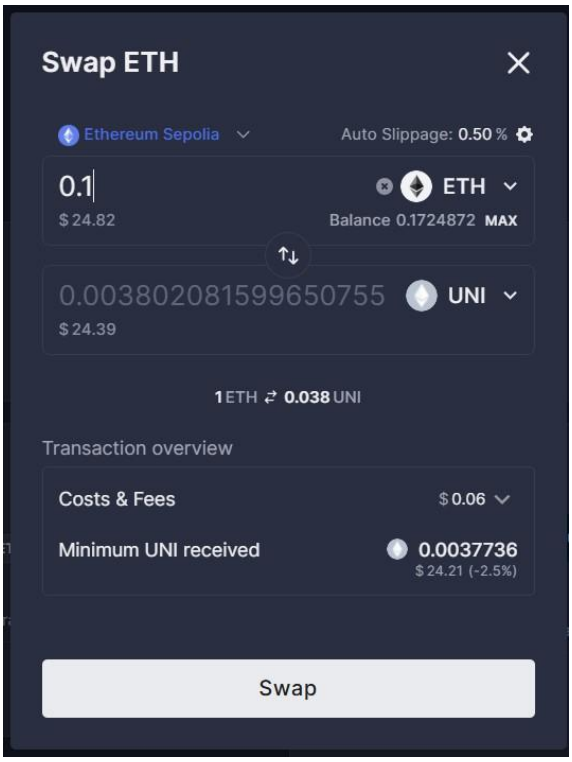


Fig3 : swap



Observation

- Token contract deployed successfully.
- Contract address was visible and verified through Avee.net.
- Initial supply was credited to the deployer's wallet (MetaMask).
- Able to transfer tokens and check balances on Avee.net interface.

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 Faculty: ***Signature of the Student:***

Name :No. :