**Unit 4 – Blockchain and Decentralized Applications**

**Lab 7 Manual**

**Objective:**

By the end of this lab, students will be able to:

* Install and configure MetaMask for a local blockchain.
* Use MetaMask to manage accounts and tokens.
* Understand how to use EtherScan to verify transactions and contracts.

**7.1. MetaMask Installation and Configuration**

**Step 1: Install MetaMask**

1. Open your browser (Chrome, Firefox, Edge, or Brave).
2. Visit [MetaMask official website](https://metamask.io) and install the browser extension.
3. Pin MetaMask to your toolbar for easy access.

**Step 2: Create a Wallet**

1. Open MetaMask and create a new wallet.
2. Set a strong password.
3. Back up the **Secret Recovery Phrase** securely. Do not share it with anyone.

**Step 3: Configure MetaMask for Ganache**

1. Open Ganache and start the blockchain. Note the **RPC Server URL** (e.g., http://127.0.0.1:7545) and the **Chain ID** (usually 1337).
2. In MetaMask:
   * Click **Settings > Networks > Add a network**.
   * Enter the following details:
     + **Network Name**: Ganache Local
     + **RPC URL**: http://127.0.0.1:7545
     + **Chain ID**: 1337
     + **Currency Symbol**: ETH
   * Save and switch to the new network.

**Step 4: Import an Account**

1. Copy a private key from any account in Ganache.
2. In MetaMask, click **Import Account** and paste the private key.

**7.2. Managing Assets in MetaMask**

**Step 1: Access MetaMask Portfolio**

1. Open MetaMask and click on the **Portfolio** button.
2. View your asset summary and manage tokens.

**Step 2: Add a Custom Token:** Use MetaMask to add a custom token (you will deploy a token in the next section).

**7.3. Exploring EtherScan**

**Step 1: Visit EtherScan:** Open [EtherScan](https://etherscan.io) (for the main Ethereum network) or explore your local blockchain equivalent for Ganache.

**Step 2: Check Wallet Transactions**

1. Copy your MetaMask wallet address.
2. Paste it into the search bar on EtherScan to view your balance and transaction history.

**Step 3: Explore Contracts:** EtherScan can also verify contract details (useful after deployment).

**7.4. Deploying a Smart Contract with Truffle**

**Step 1: Set Up the Project:** Create a new Truffle project:

**Step 2: Write a Token Contract**

1. Create a new file MyToken.sol in the contracts/ folder:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract MyToken {

string public name = "Lab6Token";

string public symbol = "L6T";

uint8 public decimals = 18;

uint256 public totalSupply;

mapping(address => uint256) public balanceOf;

event Transfer(address indexed from, address indexed to, uint256 value);

constructor(uint256 \_initialSupply) {

totalSupply = \_initialSupply \* 10 \*\* uint256(decimals);

balanceOf[msg.sender] = totalSupply;

}

function transfer(address \_to, uint256 \_value) public returns (bool success) {

require(balanceOf[msg.sender] >= \_value, "Insufficient balance");

balanceOf[msg.sender] -= \_value;

balanceOf[\_to] += \_value;

emit Transfer(msg.sender, \_to, \_value);

return true;

}

}

**Step 3: Configure Truffle**

1. Edit truffle-config.js to include:

module.exports = {

networks: {

development: {

host: "127.0.0.1",

port: 7545,

network\_id: "\*", // Match any network ID

},

},

compilers: {

solc: {

version: "^0.8.0",

},

},

};

**Step 4: Deploy the Contract**

1. Edit migrations/2\_deploy\_contracts.js:

const MyToken = artifacts.require("MyToken");

module.exports = function (deployer) {

deployer.deploy(MyToken, 1000000); // Deploy with 1,000,000 tokens

};

1. Deploy the contract

**Step 5: Interact with the Contract: truffle console --network development**

Run these commands:

let token = await MyToken.deployed();

let balance = await token.balanceOf("YOUR\_METAMASK\_ADDRESS");

console.log(balance.toString()); // Check balance

await token.transfer("RECIPIENT\_ADDRESS", 1000); // Transfer tokens

**7.5. Verify Transactions**

Step 1: Verify Transactions on EtherScan

1. Copy the transaction hash from the Truffle console.
2. View the transaction on EtherScan or the Ganache local equivalent.

Step 2: Use the Contract ABI: Use the contract's ABI to interact with its functions on the blockchain explorer.