Assignment 004

Step 1: Development Environment Setup

• Install VS Code

Step 2: Create ERC20 Smart Contract

Create a Solidity smart contract that:

• Defines token name, symbol, decimals, and total supply

• Allows token transfer between accounts

• Maintains balances using mappings

• Emits events for transparency

Step 3: Solidity Code (ERC20Token.sol)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
/*
Simple ERC20 Token Implementation
*/
contract MyERC20Token {
// Token details
string public name = "MyToken";
string public symbol = "MTK";
uint8 public decimals = 18;
uint public totalSupply;
// Mapping to store balances
mapping(address => uint) public
balanceOf;
// Event to log transfers
event Transfer(address indexed from,
address indexed to, uint value);
```

```
// Constructor runs once during
deployment
constructor(uint initialSupply) {
totalSupply = initialSupply * (10 **
uint(decimals));
balanceOf[msg.sender] = totalSupply;
}
// Transfer function
function transfer(address to, uint
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;
}
}
```

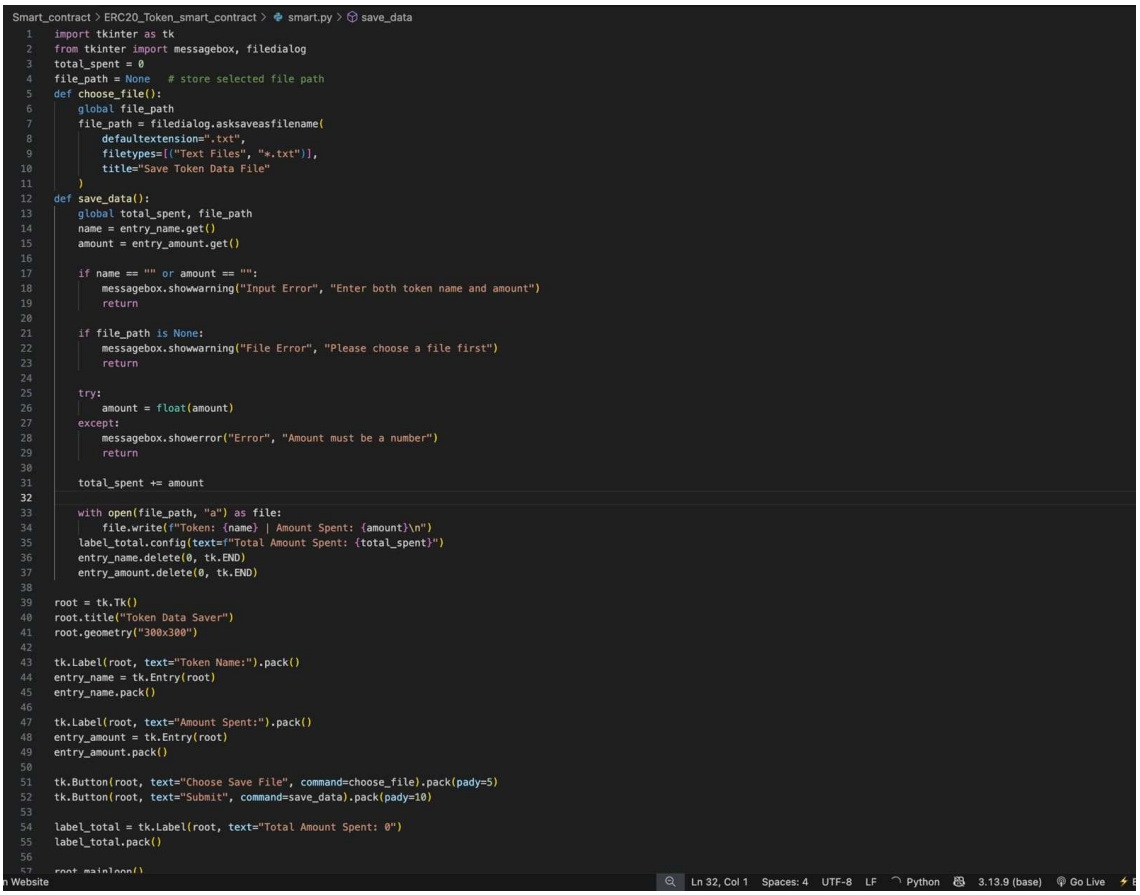
Step 4: Explanation of Logic

- mapping(address => uint) stores token balances
- constructor() assigns total supply to deployer
- transfer() moves tokens between accounts
- require() ensures sender has enough balance
- emit Transfer() logs transactions on blockchain

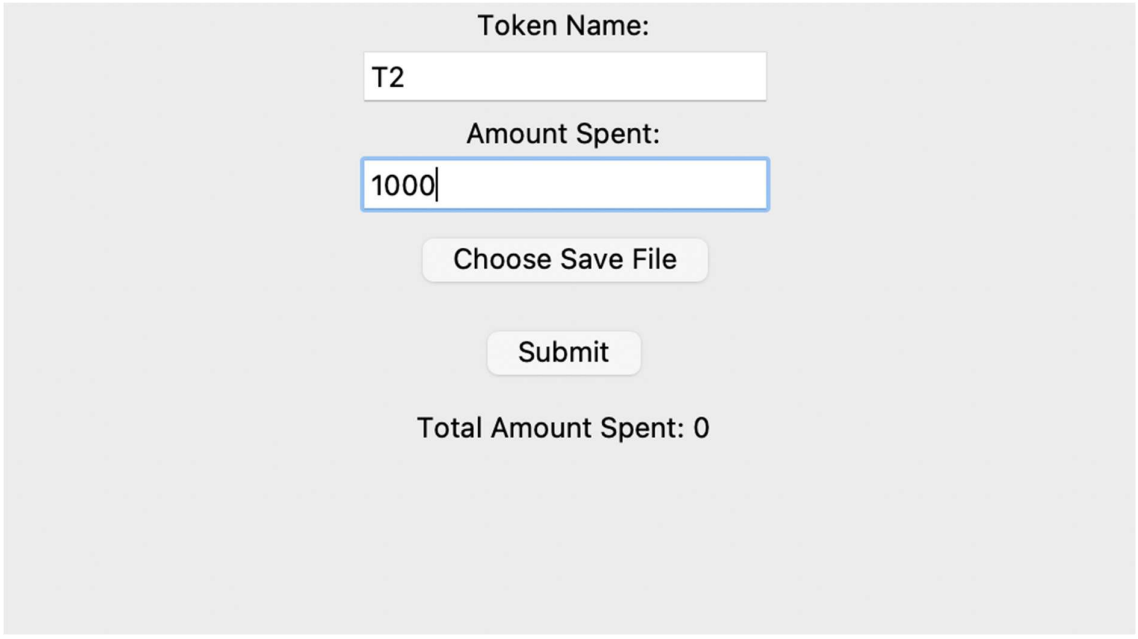
Step 5: Deployment

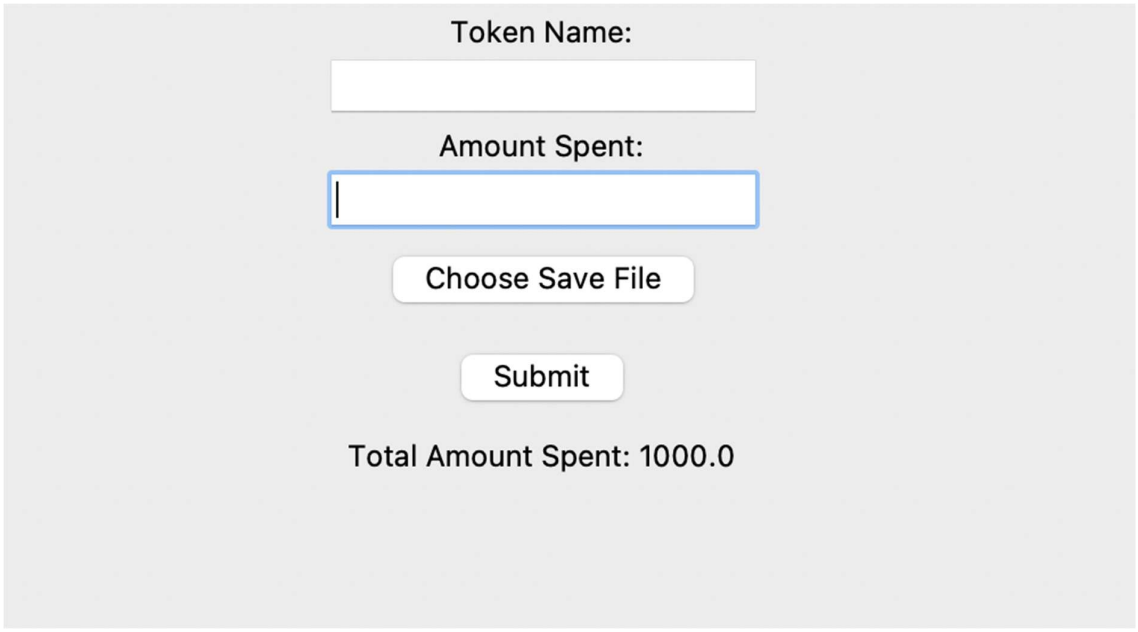
- Compile the contract using Solidity Compiler
- Deploy using Remix VM (London) or MetaMask

- Provide initial supply during deployment (e.g., 1000)



Output :-


Token Name:
T2
Amount Spent:
1000
Choose Save File
Submit
Total Amount Spent: 0


Token Name:
Amount Spent:
Choose Save File
Submit
Total Amount Spent: 1000.0