

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)

```

Smart_contract > ERC20_Token_smart_contract > smart.py > save_data
 1  import tkinter as tk
 2  from tkinter import messagebox, filedialog
 3  total_spent = 0
 4  file_path = None # store selected file path
 5  def choose_file():
 6      global file_path
 7      file_path = filedialog.askopenfilename(
 8          defaultextension=".txt",
 9          filetypes=[("Text Files", "*.txt")],
10          title="Save Token Data File"
11      )
12  def save_data():
13      global total_spent, file_path
14      name = entry_name.get()
15      amount = entry_amount.get()
16
17      if name == "" or amount == "":
18          messagebox.showwarning("Input Error", "Enter both token name and amount")
19          return
20
21      if file_path is None:
22          messagebox.showwarning("File Error", "Please choose a file first")
23          return
24
25      try:
26          amount = float(amount)
27      except:
28          messagebox.showerror("Error", "Amount must be a number")
29      return
30
31      total_spent += amount
32
33      with open(file_path, "a") as file:
34          file.write(f"Token: {name} | Amount Spent: {amount}\n")
35      label_total.config(text=f"Total Amount Spent: {total_spent}")
36      entry_name.delete(0, tk.END)
37      entry_amount.delete(0, tk.END)
38
39  root = tk.Tk()
40  root.title("Token Data Saver")
41  root.geometry("300x300")
42
43  tk.Label(root, text="Token Name:").pack()
44  entry_name = tk.Entry(root)
45  entry_name.pack()
46
47  tk.Label(root, text="Amount Spent:").pack()
48  entry_amount = tk.Entry(root)
49  entry_amount.pack()
50
51  tk.Button(root, text="Choose Save File", command=choose_file).pack(pady=5)
52  tk.Button(root, text="Submit", command=save_data).pack(pady=10)
53
54  label_total = tk.Label(root, text="Total Amount Spent: 0")
55  label_total.pack()
56
57  root.mainloop()

```

Ln 32, Col 1 Spaces: 4 UTF-8 LF ⌂ Python 3.13.9 (base) ⌂ Go Live ⌂

Output :-

Token Name:

Amount Spent:

Total Amount Spent: 0

Token Name:

Amount Spent:

Total Amount Spent: 1000.0