Blockchain Practices

Assignment 4: Tokenization, Security and Advanced DApp Development

1. Objective

The objective of this assignment is to develop an advanced decentralized application (DApp) that demonstrates practical understanding of:

- ERC-20 token standards
- Smart contract security principles
- Web3-based interaction with blockchain
- Token-based logic and access control
- Deployment and testing in a real blockchain environment

This assignment builds upon the knowledge gained in previous assignments and focuses on tokenization, secure contract design, and real-world blockchain interaction, preparing students for the final examination and real industry scenarios.

2. Technical Requirements

The project must include the following components:

2.1 Smart Contract Development (ERC-20 Token)

Students must implement a custom ERC-20 token using Solidity ^0.8.x.

Mandatory Requirements:

- Use OpenZeppelin ERC-20 standard
- Implement:
    - totalSupply
    - balanceOf
    - transfer
    - approve
    - transferFrom
- Implement ownership control using:
    - Ownable
    - onlyOwner modifier

- Add at least:
    - One minting function
    - One burning function
- Emit events:
    - TokensMinted
    - TokensBurned

The token may represent:

- Reward token
- Educational token
- Loyalty token
- Utility token

2.2 Business Logic and Access Control

Your contract must implement the following logic:

- Only the owner can mint new tokens
- Users can transfer tokens between accounts
- Token holders can approve spending limits
- Transfers must be validated using require()
- Optional (bonus):
    - Pause / unpause token transfers
    - Token supply cap

2.3 Web3 & Frontend Integration

A frontend interface must be created using:

- HTML + JavaScript
  OR
- React (optional)

The frontend must include:

- MetaMask connection
- Display of:
    - Connected wallet address

    - Token balance
- Ability to:
    - Transfer tokens
    - Approve token spending
- Display:
    - Transaction hash
    - Transaction success / failure status

Allowed libraries:

- Web3.js
- Ethers.js

2.4 Security Analysis

Students must analyze the security aspects of their implementation.

The report must include:

- Description of possible vulnerabilities:
    - Reentrancy
    - Integer overflow / underflow
    - Access control vulnerabilities
    - Front-running
- Explanation of:
    - How these risks were mitigated
    - What security mechanisms were used
- Explanation of:
    - What would happen if onlyOwner or require() checks were removed

2.5 Deployment

The smart contract must be deployed to:

- Sepolia or Goerli testnet

The following must be provided:

- Contract address
- Transaction hash

- Network name
- Deployment tool used (Remix / Hardhat)
- Screenshot of:
    - Successful deployment
    - Successful token transaction

3. System Architecture

The DApp must follow a three-layer architecture:

1. Frontend Layer

- User interface
- MetaMask integration
- Transaction interaction

2. Web3 Layer

- Web3 / Ethers.js
- Blockchain communication
- Transaction signing

3. Blockchain Layer

- ERC-20 Smart Contract
- Token storage
- Business logic execution