Centurion
UNIVERSITY
Shaping Lives...
Empowering Communities...

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 : Smart Libraries – Libraries and Proxy Contracts

Objective/Aim:

To understand the concept of **Smart Contract Libraries** and **Proxy Contracts** in Solidity and demonstrate how they are used to write modular, upgradeable, and gas-efficient smart contracts in blockchain development.

Apparatus/Software Used:

1. Visual Studio Code (VS Code)
2. MetaMask Wallet (connected to Testnet)
3. Hardhat / Truffle Framework
4. Solidity Compiler
5. OpenZeppelin Library
6. Internet Connection

Theory/Concept:

In Solidity, **libraries** and **proxy contracts** are essential for writing efficient, maintainable, and upgradeable smart contracts.

Key Concepts:

- **Libraries:**
  Libraries are reusable pieces of Solidity code that contain functions to perform common operations.
  - They reduce code duplication and gas costs.
  - Can be **linked** to contracts or used via the **using for** directive.
- **Proxy Contracts:**
  Proxy contracts act as intermediaries between users and the logic contract. Instead of redeploying an entire smart contract for updates, developers can simply deploy a new logic contract and point the proxy to it using a **delegatecall** mechanism.
- **Delegatecall:**
  A low-level function that executes code from another contract (logic contract) in the context of the proxy's storage.
  - Keeps the data in the proxy.
  - Updates logic without migrating data.
- **Advantages:**
  - Enables upgradeability of contracts.
  - Saves gas by reusing deployed libraries.
  - Simplifies contract maintenance.
  - Enhances security and modular design.

Procedure

Step 1: Set Up Development Environment

- Install Hardhat and initialize a new project:

```
npm install --save-dev hardhat
npx hardhat
```

- Create folders:
  - `/contracts` for Solidity files
  - `/scripts` for deployment scripts

Step 2: Create a Library

- Example: `MathLib.sol`

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

library MathLib {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        return a + b;
    }
}
```

Step 3: Use Library in a Contract

- Example: `Calculator.sol`

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "./MathLib.sol";

contract Calculator {
    using MathLib for uint256;

    uint256 public result;

    function calculate(uint256 a, uint256 b) public {
        result = a.add(b);
    }
}
```

Step 4: Create Proxy and Logic Contracts

- **Logic Contract (LogicV1.sol):**

```
pragma solidity ^0.8.0;
contract LogicV1 {
    uint256 public num;
    function updateNum(uint256 _num) public {
        num = _num;
    }
}
```

Proxy Contract (Proxy.sol):

```
pragma solidity ^0.8.0;
contract Proxy {
    address public implementation;
    constructor(address _impl) {
        implementation = _impl;
    }

    fallback() external payable {
        (bool success, ) = implementation.delegatecall(msg.data);
        require(success, "Delegatecall failed");
    }
}
```

Step 5: Deploy and Test

- Deploy the `LogicV1` contract and then deploy the `Proxy` contract with the logic address as the implementation.
- Interact with the proxy using the logic's ABI to verify that data updates occur through delegatecall.

Step 6: Upgrade Logic

- Deploy `LogicV2` with modified logic (e.g., multiplication instead of addition).
- Update the proxy's implementation address.
- Re-test to confirm that the proxy maintains state but now uses new logic.

Observation Table:

Component	Functionality	Result/Observation
Library (MathLib)	Reusable math operations	Performed addition efficiently
Proxy Contract	Delegated function execution	Successfully redirected to logic contract
Logic Upgrade	Updated logic via new address	Proxy preserved data after upgrade
Delegatecall Test	State retention verification	Data persisted without redeployment

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Interpretation Result and	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Faculty:

Signature of the Student:

Name

Regn.No.