



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 Experiment : **Smart Libraries – Libraries and Proxy Contracts**

### \* Coding Phase: Pseudo Code / Flow Chart / Algorithm

#### ◆ Introduction

Smart Libraries in Solidity are reusable blocks of code that help reduce redundancy, improve efficiency, and make smart contracts modular. They allow developers to define commonly used functions that can be linked to multiple contracts.

Proxy Contracts, on the other hand, enable **upgradable smart contracts** by separating logic and storage — allowing updates to the contract's logic without changing its address or data.

Open smart contract address to view code and functions.

#### ◆ Algorithm

1. **Start** the Solidity environment (Remix IDE).
2. **Create** a library with reusable functions (e.g., mathematical operations or utility logic).
3. **Import and link** the library to a main contract for function reuse.
4. **Implement** a Proxy Contract pattern with two components:
  - *Logic Contract* – contains business logic.
  - *Proxy Contract* – forwards calls and maintains data storage.
5. **Deploy** both contracts and **test** upgrades by redeploying the logic contract while keeping the proxy address same.
6. **Verify** the functionality and storage persistence.
7. **End.**

### \* Softwares used

- Chrome web browser  
Talentica.com  
<https://www.talentica.com/blogs/implementing-upgradeable-smart-contracts-using-proxy-patterns/>

**• Comprehensive Smart Contract Testing:**

Conducted in-depth validation of smart contracts to detect potential issues such as reentrancy, integer overflow, and unauthorized access using both manual and automated testing methods.

**• Code Integrity & Security Assurance:**

Ensured robust contract performance by verifying logical flow, transaction safety, and data accuracy before deployment to prevent vulnerabilities.

**• Standardized QA Framework:**

Implemented Ethereum-based testing standards using tools like **Slither**, **Mythril**, and **Remix**, maintaining consistency and transparency across all quality checks.

**• Scalable Testing Environment:**

Developed a flexible and reusable testing setup allowing re-validation, modular testing, and continuous improvement for long-term reliability and learning.

**\* Observations**

- Smart contract testing ensures secure and error-free deployment by identifying logical and security flaws before execution.
- Tools like **Remix**, **Slither**, and **Mythril** simplify vulnerability detection and code validation through automated analysis.
- Continuous testing and re-validation improve reliability, transparency, and trust in blockchain-based applications.

**ASSESSMENT**

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

*Signature of the Student:*

Name :

Regn. No. :

*Signature of the Faculty:*

Page No.....

*\*As applicable according to the experiment.  
Two sheets per experiment (10-20) to be used.*