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**

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

Smart Libraries in Solidity are reusable code modules designed to enhance efficiency, minimize redundancy, and promote modularity in smart contract development. They enable developers to create collections of commonly used functions that can be shared across multiple contracts, improving maintainability and reducing code repetition.

Proxy Contracts, on the other hand, are used to make smart contracts **upgradable** by separating the contract's **logic layer** from its **storage layer**. This design allows developers to modify or enhance the logic of a contract without changing its address or losing existing data, ensuring smooth upgrades and long-term flexibility in decentralized applications.

Algorithm

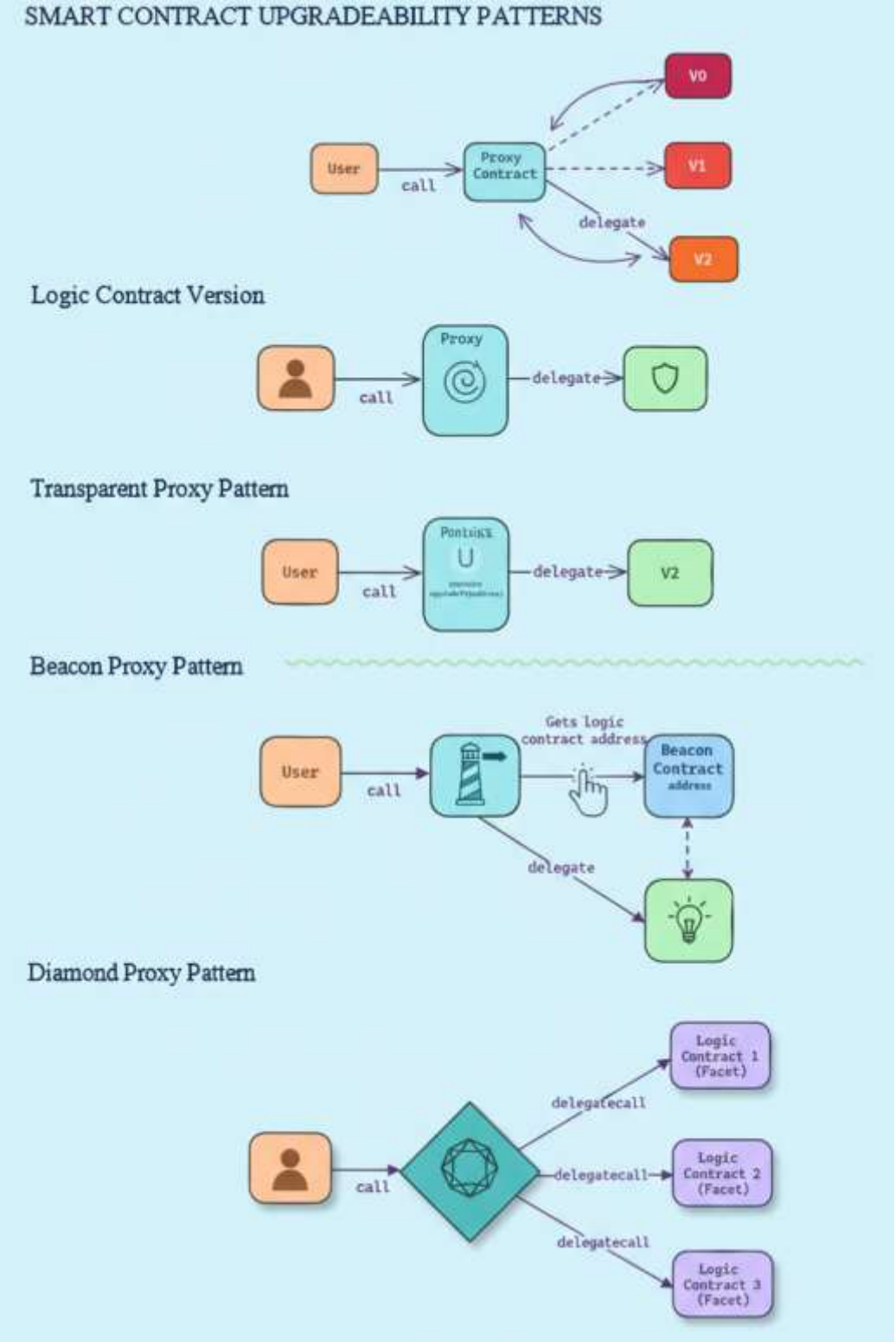
1. Launch the Solidity development environment (e.g., Remix IDE).
2. Create a library containing reusable functions such as mathematical calculations or utility operations.
3. Import and link the library within the main contract to utilize its predefined functions.
4. Design a Proxy Contract architecture consisting of two key components:
    - **Logic Contract** – defines the core business logic and functionalities.
    - **Proxy Contract** – manages data storage and delegates calls to the logic contract.
5. Deploy both the proxy and logic contracts to the blockchain.
6. Test the upgrade process by redeploying a new version of the logic contract while retaining the same proxy address to preserve stored data.
7. Confirm that the upgraded logic works correctly and that existing storage values remain unchanged.
8. End.

* Softwares used

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

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

* Implementation Phase: Final Output (no error)



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

* Implementation Phase: Final Output (no error)

• **Comprehensive Smart Contract Testing:**
Performed thorough validation of smart contracts to identify and resolve potential vulnerabilities such as **reentrancy attacks**, **integer overflows**, and **unauthorized access**. Utilized a combination of **manual inspection** and **automated testing tools** to ensure contract reliability and security.

• **Code Integrity and Security Assurance:**
Verified the **logical correctness**, **transaction safety**, and **data consistency** of smart contracts before deployment. Ensured that all functions executed as intended, preventing exploitable loopholes and maintaining overall system integrity.

• **Standardized Quality Assurance Framework:**
Adopted **Ethereum testing standards** by integrating advanced tools like **Slither**, **Mythril**, and **Remix IDE** for static and dynamic analysis. This approach ensured uniformity, transparency, and accountability throughout the entire quality assurance process.

• **Scalable and Reusable Testing Environment:**
Designed a **flexible testing framework** that supports re-validation, modular testing, and iterative improvements. This setup enhances scalability, promotes continuous learning, and ensures the long-term reliability of smart contract systems.

* 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		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

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