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: Audit 101 – Smart Contract Vulnerabilities

*Coding Phase: Pseudo Code / Flow Chart / Algorithm

1. Start
2. Open Remix IDE and create a new Solidity file.
3. Write or import a sample smart contract.
4. Compile and deploy the contract on a local or test network.
5. Analyze the code manually for common vulnerabilities.
6. Use security tools (e.g., MythX or Slither) to perform static analysis.
7. Identify and document all detected issues.
8. Apply recommended fixes or mitigation techniques.
9. Re-compile and re-test to confirm no vulnerabilities remain.
10. End

* Software used:

- Remix IDE
- MetaMask Wallet(for testnet deployment)
- Ethereum sepolia testnet
- Etherscan (testnet)

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

* Implementation Phase: Final Output (no error)

- Compile code in Remix to detect compiler-level warnings.

- Run automated tools (MythX / Slither) to scan for security risks.

- Review logic manually for misuse of access modifiers or unsafe calls

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

contract VulnerableBank {
    mapping(address => uint256) public balances;

    function deposit() public payable {    infinite gas
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint256 amount) public {    infinite gas
        require(balances[msg.sender] >= amount, "Insufficient balance");
        (bool success, ) = msg.sender.call{value: amount}("");
        require(success, "Transfer failed");
        balances[msg.sender] -= amount;
    }

    function getBalance() public view returns (uint256) {    2496 gas
        return balances[msg.sender];
    }
}
```

Fixes Applied:

- Apply fixes such as adding modifiers, SafeMath, or Guards.

- Re-deploy contract and re-test using Remix and Ganache.

- Verify no remaining warnings or vulnerabilities.

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

contract SecureBank {
    mapping(address => uint256) public balances;

    function deposit() external payable {    infinite gas
        require(msg.value > 0, "Must deposit non-zero amount");
        balances[msg.sender] += msg.value;
    }

    function getBalance() external view returns (uint256) {    2496 gas
        return balances[msg.sender];
    }
}
```

* Observation:

- Auditing revealed several potential vulnerabilities in logic and access control.
- After mitigation, the smart contract executed securely with no reentrancy or overflow issues.
- Security auditing is essential before deploying any contract to a live blockchain.

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 :

Signature of the Faculty :

Regn. No. :

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