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 : Contract QA – Testing Smart Contracts

Objective/Aim:

To perform Quality Assurance (QA) testing on a deployed smart contract by validating its core functions such as payment creation, release, refund, and timeout handling. This includes interacting with the contract using MetaMask, a testnet environment, and monitoring transactions to ensure correct behavior.

Apparatus/Software Used:

- MetaMask Wallet
- Brave/Chrome Browser
- Remix IDE or Hardhat Environment
- Ethereum Sepolia Testnet
- Etherscan (for transaction verification)
- Testnet ETH from faucet

Theory/Concept:

Smart contract QA ensures that decentralized applications behave correctly and securely once deployed on a blockchain. Since smart contracts are immutable, proper testing prevents financial loss, security flaws, and logic failures.

Key Concepts:

Functional Testing:
Validates whether functions such as createPaymentETH(), releasePayment(), refundPayment(), and triggerRefundAfterTimeout() behave as expected.

Security Testing:
Ensures the contract is protected from:

- Reentrancy attacks
- Unauthorized access
- Overflow/underflow
- Incorrect state changes

Gas Estimation:
Every smart contract execution requires gas; QA includes checking if operations use reasonable gas.

Procedure:

1. Setup Environment
   - Open MetaMask and switch to the Sepolia Testnet.
   - Load the deployed smart contract inside Remix or connect via Hardhat scripts.
2. Testing Payment Creation
   - Connect MetaMask to Remix.
   - Select the createPaymentETH() function.
   - Enter seller address, timeout value, and metadata.
   - In the value field, enter test ETH to send.
   - Click Transact and confirm in MetaMask.
   - Wait for transaction confirmation.

3. Testing Release of Payment
   - Using the seller's MetaMask account, call releasePayment(id).
   - Confirm the transaction and verify that the seller receives the net amount (minus fee).
4. Testing Refund Before Timeout
   - Switch back to payer account.
   - Call refundPayment(id) before the timeout expires.
   - Check that the full amount is returned to the payer.
5. Testing Timeout Refund
   - Create another payment with a short timeout (for testing).
   - Wait until the timeout expires.
   - Call triggerRefundAfterTimeout(id) from any address.
   - Verify that funds return to the payer.
7. Etherscan Verification
   - Open Etherscan Testnet Explorer.
   - Review transaction details such as:
   - Gas usage
   - Event logs
   - Status (success/failure)

Observation

Test Case	Action Performed	Expected Outcome	Result
Payment Creation	createPaymentETH()	Contract stores escrow payment	✅ Success
Release Payment	releasePayment()	Seller receives net funds	✅ Success
Refund Payment	refundPayment()	Full amount returned to payer	✅ Success
Timeout Refund	triggerRefundAfterTimeout()	Auto-refund after expiry	✅ Success
Cancel Payment	cancelPayment()	Refund if no timeout	✅ Success
Security Check	Reentrancy attempt	Blocked by ReentrancyGuard	✅ Verified

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: