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 Experiment: Contract QA – Testing Smart Contracts

*Aim :

To understand and perform Quality Assurance (QA) testing for smart contracts deployed on blockchain platforms (like Ethereum), ensuring their correctness, reliability, and security.

*Objective:

- ➢ ☐ To learn how to test smart contracts using QA methodologies.
- ➢ ☐ To identify and fix logical or security issues before deployment.
- ➢ ☐ To use simulation tools and test networks for verification.

*Coding Phase: Pseudo Code / Flow Chart / Algorithm

1. Open Remix IDE and create a new Solidity smart contract (e.g., SimpleStorage.sol).
2. Write a simple contract with basic functions (store, retrieve, modify).
3. Compile the smart contract to check for syntax or compilation errors.
4. Deploy the contract on a test network (e.g., Sepolia).
5. Perform different QA test cases:

- ➢ Positive Testing: valid inputs and expected outcomes.
- ➢ Negative Testing: invalid inputs, edge cases, gas limit checks.
- ➢ Security Testing: reentrancy, integer overflow/underflow, access control.

6. Observe output logs and execution gas.
7. Analyze results and confirm if contract logic performs as intended.
8. Document errors and apply fixes.
9. Re-test until all test cases pass.

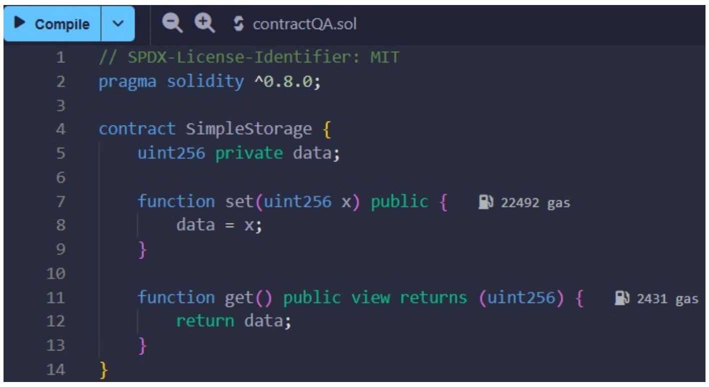
***** *As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.

* Software used:

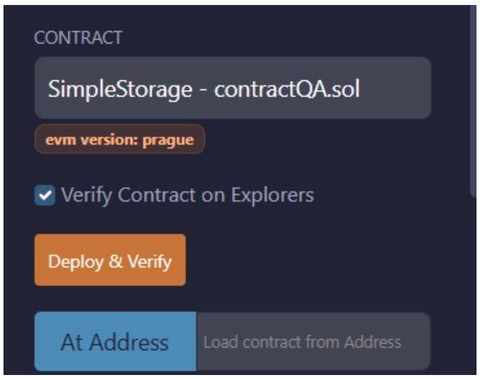
- ➢ Remix IDE
- ➢ Ethereum sepolia testnet
- ➢ Etherscan (testnet)

* Implementation Phase: Final Output (no error)

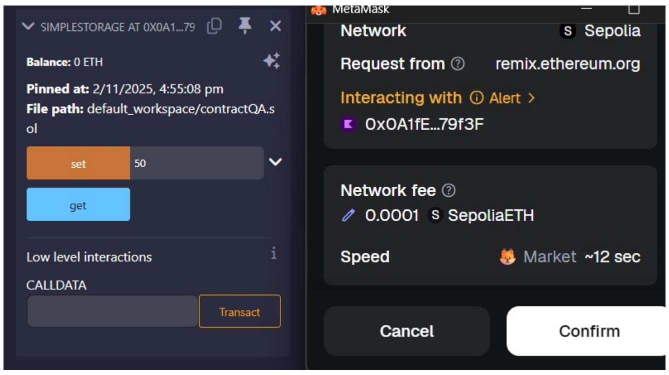
- ➢ Open the Remix IDE and write the smart contract.
- ➢ Compile the contract.
- ➢ Deploy using the “Injected Provider-MetaMask” environment.
- ➢ In the deployed contract tab:
- ➢ Use set() to store a value.
- ➢ Use get() to retrieve and verify.
- ➢ Confirm that the function outputs are correct.



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

contract SimpleStorage {
    uint256 private data;

    function set(uint256 x) public {    22492 gas
        data = x;
    }

    function get() public view returns (uint256) {    2431 gas
        return data;
    }
}
```

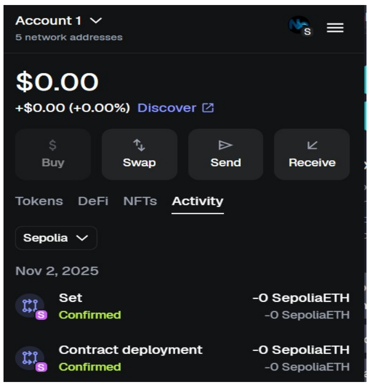


SimpleStorage - contractQA.sol
evm version: prague
✓ Verify Contract on Explorers



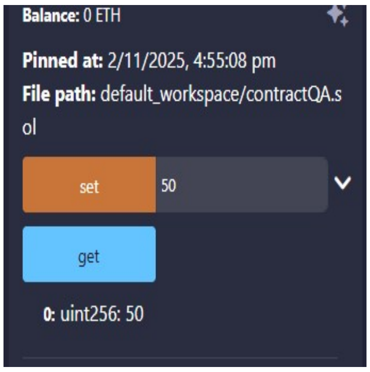
Balance: 0 ETH
Pinned at: 2/11/2025, 4:55:08 pm
File path: default_workspace/contractQA.sol
set 50

Network: Sepolia
Request from: remix.ethereum.org
Interacting with: Alert
0x0A1fE...79f3F
Network fee: 0.0001 SepoliaETH
Speed: Market ~12 sec

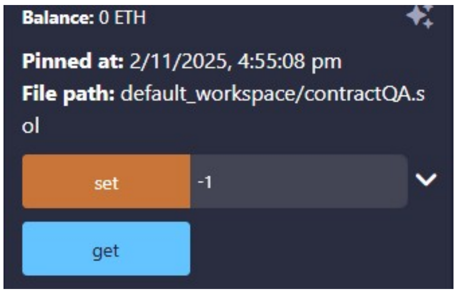


$0.00
+$0.00 (+0.00%)

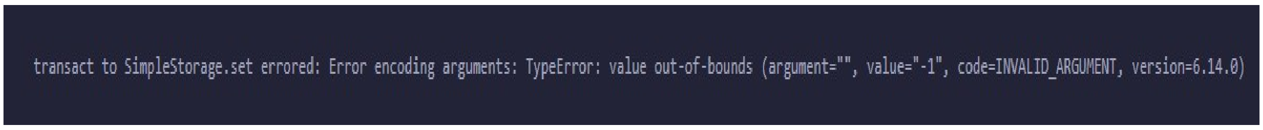
Nov 2, 2025
Set — Confirmed — -0 SepoliaETH / -0 SepoliaETH
Contract deployment — Confirmed — -0 SepoliaETH / -0 SepoliaETH



Balance: 0 ETH
Pinned at: 2/11/2025, 4:55:08 pm
File path: default_workspace/contractQA.sol
set 50
0: uint256: 50



Balance: 0 ETH
Pinned at: 2/11/2025, 4:55:08 pm
File path: default_workspace/contractQA.sol
set -1



```
transact to SimpleStorage.set errored: Error encoding arguments: TypeError: value out-of-bounds (argument="", value="-1", code=INVALID_ARGUMENT, version=6.14.0)
```

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

* Observation:

- Smart contract testing ensures reliability and correctness before deployment.
- Logical or security bugs can be detected early through QA testing.
- Tools like Remix, MythX, and Etherscan help validate code and execution flow.

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. :

Page No…………….

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