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 understand the importance of **Quality Assurance (QA)** in smart contracts.
- To learn how to **test, debug, and verify** smart contracts before deployment.
- To implement **unit testing** using blockchain development tools.

Apparatus/Software Used:

- MetaMask
- Remix IDE
- Vs code

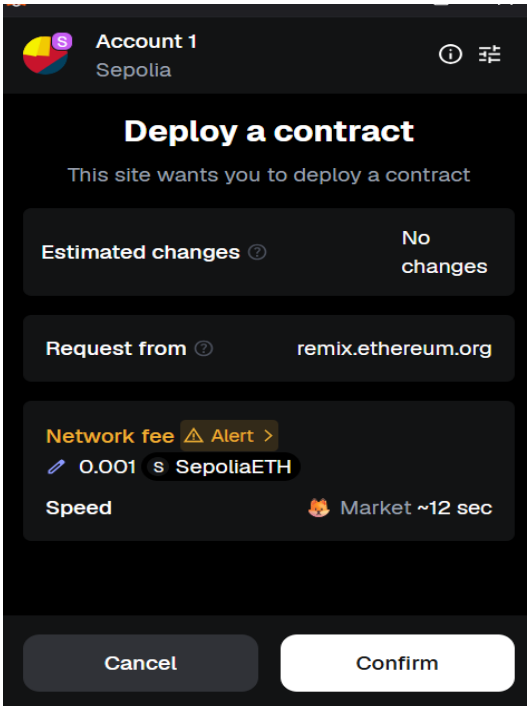
Procedure:

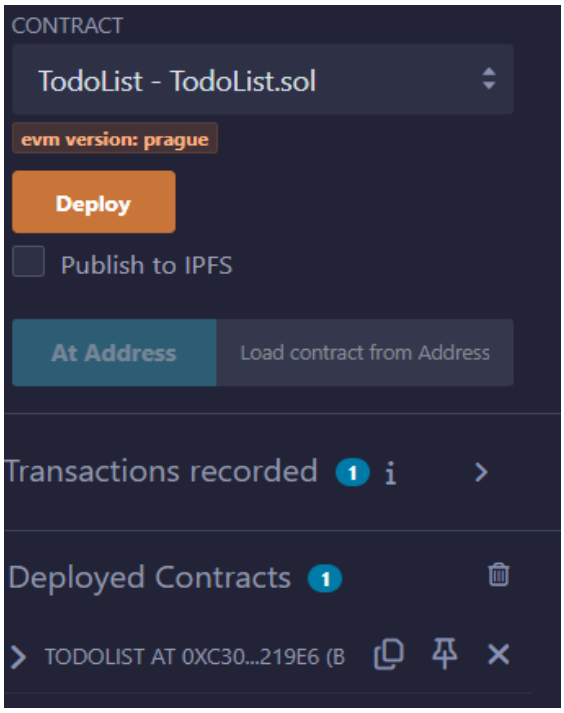
1.Open Remix IDE and write a simple TodoList.sol smart contract.

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

contract TodoList {
    // Define a Task structure with a description and completion status
    struct Task {
        string description;
        bool completed;
    }

    Task[] public tasks;  // dynamic array of Task structs

    // Add a new task to the list (description provided by user)
    function addTask(string memory _desc) public {    infinite gas
        tasks.push(Task(_desc, false));
    }

    // Mark a task as completed by index
    function markCompleted(uint _index) public {    29057 gas
        require(_index < tasks.length, "Index out of range");
        tasks[_index].completed = true;
    }

    // Get a task's details by index (returns tuple: description and completed flag)
    function getTask(uint _index) public view returns (string memory, bool) {    infinite gas
        require(_index < tasks.length, "Index out of range");
        Task storage t = tasks[_index];
        return (t.description, t.completed);
    }
```

Procedure:

2.After writing smart contract compile it and copy the abi .
3.Then go to the deploye section and delpoye it and confirm transaction on MetaMask.

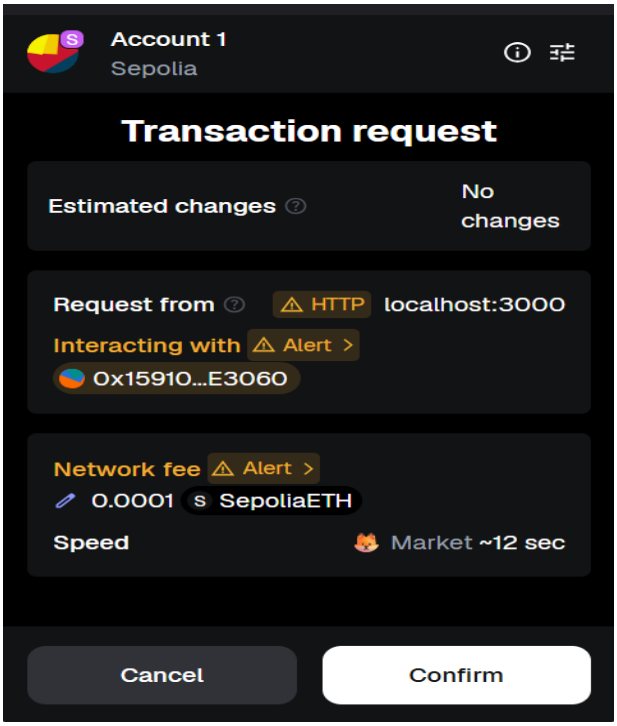




3. Now open Vs code and past the abi and write the frontend in app.js and ether.js after connecting our smart contract with frontend run the the code .

4.Now we test our functions:

- Enter a task and click **Add Task** → MetaMask will pop up for confirmation.



- After confirming, the task appears in the **task list** with (not completed).

- Mark the task as completed in Remix or extend frontend with a button → reload page → shows completed.

Observation:

- The smart contract TodoList was compiled and deployed successfully in Remix. A new task was added using the addTask function, and it appeared with status false (not completed). On calling getTask(0), the correct description and completion status were returned. After using the markCompleted(0) function, the task status changed to true (completed). The getTotalTasks function displayed the total number of tasks correctly.

- When connected with the frontend, tasks could be added from the web page and were displayed in the list with incomplete and completed. The frontend interacted smoothly with MetaMask, confirming transactions for adding tasks.

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.