

St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2025-2026

Class: BE-ITA/B, Semester: VIII

Subject: BlockChain Lab

Experiment 4

1. **Aim:** To implement smart contract using Remix IDE and deploy it using Ganache and MetaMask.

2. **Objective:** To ...

- explore working of Ganache ethernet network.
- test a smart contract by using Remix IDE

3. **Lab outcome:** After performing the experiment, the students will be able to **implement** smart contracts in Ethereum using different development frameworks (PO3, PSO2, BL3)

4. **Prerequisite:**

- Fundamental knowledge of blockchain
- Knowledge of the Ethereum platform and Remix IDE
- Familiarity with the Solidity programming language

5. **Requirements:** The following are the requirements –
Remix IDE, Ganache Provider, MetaMask wallet etc.

6. Pre-Experiment Theory:

What is Ganache?

Ganache is a private Ethereum blockchain development environment that allows to you emulate the Ethereum blockchain so that you can interact with smart contracts in your own private blockchain. You can use Ganache across the entire development cycle; enabling you to develop, deploy, and test your dApps (decentralized Applications) in a safe and deterministic environment.

7. Laboratory Exercise

A. Steps to be implemented.

To Follow the procedure given below to build smart contract in Remix IDE

- Download and install Ganache from <https://truffleframework.com/ganache>.
- Download and install the official MetaMask extension/ plugin.
- Open Remix IDE in Google Chrome.
- Write smart contract by creating new file under contract folder, with .sol extension.
- Write your contract code using solidity language.
- Click on compiler icon, choose compiler version or keep default and click on compile button.
- For deployment...

Part A) First start Ganache provider and then in Remix IDE, choose environment Ganache Provider and then deploy the said contract.

Part B) Set-up Ganache with MetaMask and then in Remix IDE choose environment Injected Provider- MetaMask and then deploy the said contract.

- Under deployed contract, get the output of your contract.

B. Program Code

Write a smart contract StudentMarks.sol to create student database with attributes, id, name, dept, sub1marks, sub2Marks etc. and deploy it using Ganache and MetaMask.

8. Post Experimental Exercise-**A. Questions:**

1. List down the details of Ganache IDE
2. Create any smart contract that you have not done so far and deploy it.

B. Results/Observations/Program output:

Present the program input/output results if any and comment on the same.

C. Conclusion:

Write what was performed in the experiment.

Write which tools you used to perform the experiment

Write what you inferred from the output obtained.

D. References:

<https://www.geeksforgeeks.org/how-to-use-metamask-to-deploy-a-smart-contract-in-solidity-blockchain/>

<https://www.geeksforgeeks.org/how-to-set-up-ganche-with-metamask/>

Mastering Ethereum, Building Smart Contract and Dapps, Andreas M. Antonopoulos Dr. Gavin Wood, O'reilly

The screenshot shows the Remix IDE interface. The left sidebar contains the 'DEPLOY & RUN TRANSACTIONS' panel with options for 'Deployed Contracts', 'STORE', and 'CALLDATA'. The main editor displays Solidity code for a 'Storage' contract. The right sidebar features the 'REMIX ASSISTANT' with a 'RemixAI' chat interface. The bottom status bar shows 'Scan Alert' and 'Initialize as git repo'.

The 'ENVIRONMENT' panel is configured for the 'Dev - Ganache Provider'. It shows the 'Geth Testnet (1337) network' and an account with address '0x277...a38cd' and a balance of '100.0 ETH'. The 'GAS LIMIT' is set to 'Estimated Gas' with a custom value of '3000000'. The 'VALUE' is set to '0 Wei'. The 'CONTRACT' is 'Storage - contracts/1_Storage.sol' with 'evm version: london'. The 'Verify Contract on Explorers' checkbox is checked. The 'Deploy & Verify' button is visible at the bottom.

The 'DEPLOY & RUN TRANSACTIONS' panel is configured for the 'Dev - Ganache Provider'. It shows the 'Geth Testnet (1337) network' and an account with address '0xfbe...a71e4' and a balance of '99.99981823235'. The 'GAS LIMIT' is set to 'Estimated Gas' with a custom value of '3000000'. The 'VALUE' is set to '0 Wei'. The 'CONTRACT' is 'Storage - contracts/1_Storage.sol' with 'evm version: london'. The 'Verify Contract on Explorers' checkbox is checked. The 'Deploy & Verify' button is visible at the bottom.

The 'DEPLOY & RUN TRANSACTIONS' panel is configured for the 'Injected Provider - MetaMask'. It shows the 'Geth Testnet (1337) network' and an account with address '0xfbe...a71e4' and a balance of '99.99643029235'. The 'GAS LIMIT' is set to 'Estimated Gas' with a custom value of '3000000'. The 'VALUE' is set to '0 Wei'. The 'CONTRACT' is 'Storage - contracts/1_Storage.sol' with 'evm version: london'. The 'Verify Contract on Explorers' checkbox is checked. The 'Deploy & Verify' button is visible at the bottom.

Ganache						
ACCOUNTS	BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS	SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 2	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MERGE	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING
BLOCK 2	MINED ON 2026-02-04 09:38:11	GAS USED 43724	1 TRANSACTION			
BLOCK 1	MINED ON 2026-02-04 09:37:54	GAS USED 125673	1 TRANSACTION			
BLOCK 0	MINED ON 2026-02-04 09:29:03	GAS USED 0	NO TRANSACTIONS			

TX HASH				CONTRACT CALL
0x1bd43f69f779e8aeb6fa60ef63541c085e22a504905800b4daed29292ed9f7bd				
FROM ADDRESS 0x277EDC51b3222D917271bf8cfB3E0dc3853A38cD	TO CONTRACT ADDRESS 0x56f7c0Eed209F04D8346D08B0B4d81970C316A206	GAS USED 43724	VALUE 0	
TX HASH				CONTRACT CREATION
0x03345a0ae0d803378e0f033644c4f7f187a3edb270241a21cb9eab53f67d0cff				
FROM ADDRESS 0x277EDC51b3222D917271bf8cfB3E0dc3853A38cD	CREATED CONTRACT ADDRESS 0x56f7c0Eed209F04D8346D08B0B4d81970C316A206	GAS USED 125673	VALUE 0	

Metamask-ganashe setup

The screenshot displays the Ganache desktop application. The top navigation bar includes tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. The BLOCKS tab is selected, showing a table of blocks. The TRANSACTIONS tab is also visible, showing a list of transactions.

BLOCK	MINED ON	GAS PRICE	GAS LIMIT	HARDWARE	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
BLOCK 4	2026-02-04 09:51:47	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	1 TRANSACTION
BLOCK 3	2026-02-04 09:51:33	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	1 TRANSACTION
BLOCK 2	2026-02-04 09:38:11	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	1 TRANSACTION
BLOCK 1	2026-02-04 09:37:54	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	1 TRANSACTION
BLOCK 0	2026-02-04 09:29:03	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	NO TRANSACTIONS

The screenshot displays the Remix IDE interface. The 'Deploy & Run' tab is active, showing the 'Storage' contract. The 'Remix Assistant' panel is open, providing guidance on building smart contracts. The 'MetaMask' extension is also visible, showing the 'Imported Account 1' and the 'Deploy a contract' button.

The screenshot displays the Ganache desktop application with the TRANSACTIONS tab selected. It shows a list of transactions with their details.

TX HASH	FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0x2d7d80f4924a35cf2281de96e076fd1d6574d60814470a060bba482aebc7f7de	0xf8e7ab6fb1aDd89805db62629608852f75a71e4	0xf66b35ffdAf40833b850191ACd9c1a712e2352B	43724	0
0x2a311972cc9ce31bbc717abd54198e4aab8f2ca22605bea62f20b19cd51a4644	0xf8e7ab6fb1aDd89805db62629608852f75a71e4	0xf66b35ffdAf40833b850191ACd9c1a712e2352B	125673	0
0x208ad3f4791edfe01988d8ed43135e4c8b29bab873832fc2c28f4e2768fd6206	0xf8e7ab6fb1aDd89805db62629608852f75a71e4	0xbcc011c8Ae9A34E07338eDcC5A3988064993e3	43724	0

Program Code-

Write a smart contract StudentMarks.sol to create student database with attributes, id,

name, dept, sub1marks, sub2Marks etc

Code-

/ SPDX-License-Identifier: MIT

uint256 _sub1Marks,

```

pragma solidity ^0.8.0;
contract StudentMarks {
    struct Student {
        uint256 id;
        string name;
        string dept;
        uint256 sub1Marks;
        uint256 sub2Marks;
    }
    Student[] private students;
    mapping(uint256 => Student) private studentById;

    function addStudent(
        uint256 _id,
        string memory _name,
        string memory _dept,
        uint256 _sub1Marks,
        uint256 _sub2Marks
    ) public {
        require(_sub1Marks <= 100 && _sub2Marks <= 100, "Marks cannot exceed 100");
        Student memory newStudent = Student({
            id: _id,
            name: _name,
            dept: _dept,
            sub1Marks: _sub1Marks,
            sub2Marks: _sub2Marks
        });
        students.push(newStudent);
        studentById[_id] = newStudent;
    }

    function getStudent(uint256 _id) public view
    returns (Student memory) {
        return studentById[_id];
    }
}

```

The screenshot displays a web browser with two main windows. The top window is the Remix IDE, showing the 'StudentMarks.sol' contract. The contract defines a 'Student' struct with fields for 'id', 'name', 'dept', 'sub1Marks', and 'sub2Marks'. It includes a private array 'students' and a private mapping 'studentById'. The 'addStudent' function is public and takes parameters for 'id', 'name', 'dept', 'sub1Marks', and 'sub2Marks'. It checks if the marks are within the limit of 100 and then adds the student to the array and mapping. The 'getStudent' function is public and view, returning the student object from the mapping. The bottom window is the MetaMask transaction confirmation screen. It shows a 'Transaction request' for the 'addStudent' function. The network is set to 'Ganache', the request is from 'remix.ethereum.org', and the interacting address is '0x469A3...cEFeF'. The network fee is 0.0054 ETH. The 'Confirm' button is visible. Below the transaction screen, the Ganache interface shows the 'BLOCK 6' details. It includes a table with columns for 'GAS USED', 'GAS LIMIT', 'MINED ON', and 'BLOCK HASH'. The values are: GAS USED: 270712, GAS LIMIT: 6721975, MINED ON: 2026-02-04 10:01:07, and BLOCK HASH: 0xca9957bc42799cb27d3fe60c89a9dd1544137433ef24b180b32951e9b9abfa8a. Below this, the 'TX HASH' is shown as '0x15aa1f01ed72f758adff4b1e28b29241c8a3467593dd941b6c6e2d733f6d6e4f'. The 'FROM ADDRESS' is '0xf8e7ab6Fb1aDd898D5db6262960B852f75a71e4' and the 'TO CONTRACT ADDRESS' is '0x469A3351E672162D604F2f8F1A86d3B97BDcEFeF'. The 'GAS USED' is 270712 and the 'VALUE' is 0.