



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 : Dive into Ethereum Clients and EVM

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

Ethereum Clients

- Ethereum clients are the fundamental components that allow computers, known as nodes, to join, communicate, and maintain the Ethereum network.
- They manage crucial blockchain functions such as propagating transactions, verifying blocks, executing smart contracts, and keeping the network synchronized.
- Each client represents a separate implementation of the Ethereum protocol, written in different programming languages to promote diversity and security.
- Commonly used clients include **Geth**, **Nethermind**, **Besu**, and **Erigon**, each offering unique features, performance optimizations, and compatibility options.
- Ethereum clients also include tools for developers to interact with the blockchain, deploy contracts, and monitor transactions, making them essential for both users and developers.

Ethereum Virtual Machine (EVM)

- The **Ethereum Virtual Machine** is the computational layer of Ethereum responsible for processing and executing all smart contracts and decentralized applications (DApps).
- It operates as a **virtual computer** distributed across thousands of nodes, ensuring that all code executes identically on every participant's machine.
- The EVM transforms human-readable smart contract code into bytecode that can be executed in a secure, deterministic, and isolated manner.
- It guarantees **deterministic computation**, where every node reaches the same output for identical input, maintaining network-wide consensus.
- All operations in the EVM require **gas**, which measures computational effort, prevents spam, and ensures efficient resource usage.
- By isolating contract execution from the host machine, the EVM protects the overall network from faulty or malicious code.
- This robust execution environment has made the EVM the **foundation for Ethereum's ecosystem**, powering innovations in **DeFi**, **NFTs**, **DAOs**, and other Web3 technologies.

* Softwares used

1. Brave browser
2. MetaMask Wallet
3. Remix IDE
4. Sepolia Testnet

* Implementation Phase: Final Output (no error)

This Solidity program is a **simple storage smart contract** that lets users save a number on the blockchain and retrieve it later.

It contains one state variable (storedNumber), a setter function (setNumber) to update the value, and a getter function (getNumber) to read the value.

```

1 // SPDX-License-Identifier: GPL-1.0
2
3 pragma solidity ^0.8.2 ^0.9.0;
4
5 /**
6  * @title Storage
7  * @dev Store & retrieve value in a variable
8  * @custom:dev-run-script ./scripts/deploy_with_ethers.ts
9  */
10 contract Storage {
11     uint256 number;
12
13     /**
14      * @dev Store value in variable
15      * @param num value to store
16      */
17     function store(uint256 num) public { 22514 gas
18         number = num;
19     }
20
21     /**
22      * @dev Return value
23      * @return value of 'number'
24      */
25     function retrieve() public view returns (uint256){ 2409 gas
26         return number;
27     }
28 }

```

✓ [block:9553149 txIndex:2] from: 0x776...feb14 to: Storage.(constructor)
value: 0 wei data: 0x608...e0033 logs: 0 hash: 0xcb5...fb42c Debug

Verification process started...

ENVIRONMENT

Injected Provider - MetaMask

Sepolia (11155111) network

ACCOUNT

0x776...feb14 (1.298026602798)

+ Create Smart Account

GAS LIMIT

☒ Estimated Gas

☐ Custom 3000000

VALUE

0 Wei

CONTRACT

Storage - contracts/1_Storage.sol

evm version: prague

☒ Verify Contract on Explorers

Deployed Contracts 1

✓ STORAGE AT 0XDD4...3E340 (BI

Balance: 0 ETH

store uint256 num

retrieve

Low level interactions

CALLDATA

Transact

* Observations:

- The Ethereum client was installed and configured successfully.
- The client synchronized with the Ethereum blockchain (testnet/mainnet as configured).
- Account creation and wallet address generation were successful.
- The client responded correctly to JSON-RPC/CLI commands.
- Transactions were executed and verified without errors.
- Logs/output confirmed the proper working of the Ethereum client.

The experiment was successfully carried out. The Ethereum client was installed, configured, and executed without errors. It synchronized with the blockchain, allowed account creation, and responded to commands correctly. This confirms that Ethereum clients are essential for interacting with the Ethereum network, validating transactions, and executing smart contracts through the EVM.

* Conclusion

The experiment was successfully carried out. The Ethereum client was installed, configured, and executed without errors. It synchronized with the blockchain, allowed account creation, and responded to commands correctly. This confirms that Ethereum clients are essential for interacting with the Ethereum network, validating transactions, and executing smart contracts through the EVM.

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. :

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