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 : Dive into Ethereum – Clients and EVM

Objective/Aim:

To study how Ethereum blockchain clients and the Ethereum Virtual Machine (EVM) work together to execute smart contracts, manage accounts, and handle computational resources through gas.

Apparatus/Software Used:

- **Programming Language: Solidity**
- **IDE/Compiler: Remix IDE**
- **Client Software: Geth, Hardhat, or Ganache**
- **Wallet: MetaMask**

Theory/Concept:

Ethereum is an open-source blockchain platform that supports **smart contracts** and **decentralized applications (dApps)**. It enables trustless computation through programmable contracts that execute automatically on the blockchain.

Ethereum Clients:

Clients are software programs that implement the Ethereum protocol, allowing computers to participate in the network. Examples include:

- **Geth (Go-Ethereum):** The most widely used official client written in Go.
- **Nethermind:** A client built in C# focused on performance and modularity.
- **Besu:** An enterprise-grade Ethereum client written in Java.

Each client helps nodes sync with the network, validate transactions, and deploy contracts.

Ethereum Virtual Machine (EVM):

The EVM is the execution engine responsible for running all smart contracts on the Ethereum blockchain. It converts smart contract code into **bytecode** and ensures deterministic execution across all nodes. Every action in the EVM consumes **gas**, a unit that measures computational cost, ensuring efficient use of network resources.

Ethereum Accounts:

- **Externally Owned Account (EOA):** Managed by a private key and used by individuals.
- **Contract Account:** Controlled by smart contract logic and code execution.

Gas Concept:

Gas represents the fee required to perform operations on the EVM. It prevents abuse of computational resources by charging Ether for each action, like storing data or calling a function.

Procedure:

1. Setup Ethereum Client

    - Install Geth or use Hardhat or Foundry to create local Etherum network
    - Initialize and run a private blockchain instance

2. Create Ethereum Accounts

    - Generate new accounts using Geth command – line interface or Metamask
    - Add test Ether to the accounts through a **testnet faucet** or **Ganache**.

3. Write and Deploy a Smart Contract on EVM

    - Write a simple Solidity smart contract (e.g., HelloWorld.sol).
    - Compile it using **Remix IDE** or **Hardhat compiler**.
    - Deploy contract on the local Ethereum client or testnet.

4. Interact with Deployed Contract

    - Use **Web3.js** or **Ethers.js** to call contract functions.\
    - Record transaction details such as gas usage and transaction hash

5. Analyze EVM Execution

    - Observe the bytecode execution and storage updates in the Ethereum client console.
    - Check logs and verify gas consumption during transaction execution

Observation Table:

The smart contract was compiled and deployed successfully on the local Ethereum client. When interacting with the contract, the **EVM** executed the code correctly, updated the contract's state variables, and displayed gas consumption for each transaction. Logs and transaction receipts confirmed that the EVM processed and validated operations as expected.

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 Faculty:

Signature of the Student:
Name :
Regn.