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 understand Ethereum architecture and its clients.
- To explore the working of the Ethereum Virtual Machine (EVM).
- To gain hands-on experience with Ethereum clients (like Geth or Hardhat/Foundry) and interact with the EVM.

Apparatus/Software Used:

- **Programming Language**: Solidity
- **Blockchain Explorer**: Remix IDE
- **Ethereum Client**: Geth / Hardhat

.

Theory concept:

Ethereum: A decentralized blockchain platform supporting smart contracts and decentralized applications (dApps).

Ethereum Clients: Software implementations of the Ethereum protocol (e.g., Geth, Nethermind, Besu). They allow nodes to join the Ethereum network, sync data, and interact with the blockchain.

Ethereum Virtual Machine (EVM): A decentralized computation engine that executes smart contracts. It uses bytecode, gas, and opcodes to ensure deterministic execution of contracts across all nodes.

Accounts:

- Externally Owned Accounts (EOA): Controlled by private keys.
- Contract Accounts: Controlled by smart contract code.

Gas: The unit that measures the amount of computational effort required to execute operations on the EVM.

Procedure:

Setup Ethereum Client

- Install Geth or use Hardhat/Foundry.
- Initialize a private Ethereum test network.

⍰ **Create Ethereum Accounts**

- Generate accounts using Geth CLI or MetaMask.
- Fund test accounts with Ether (using testnet faucet or Ganache).

⍰ **Deploy Smart Contract on EVM**

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

⍰ **Interact with Contract**

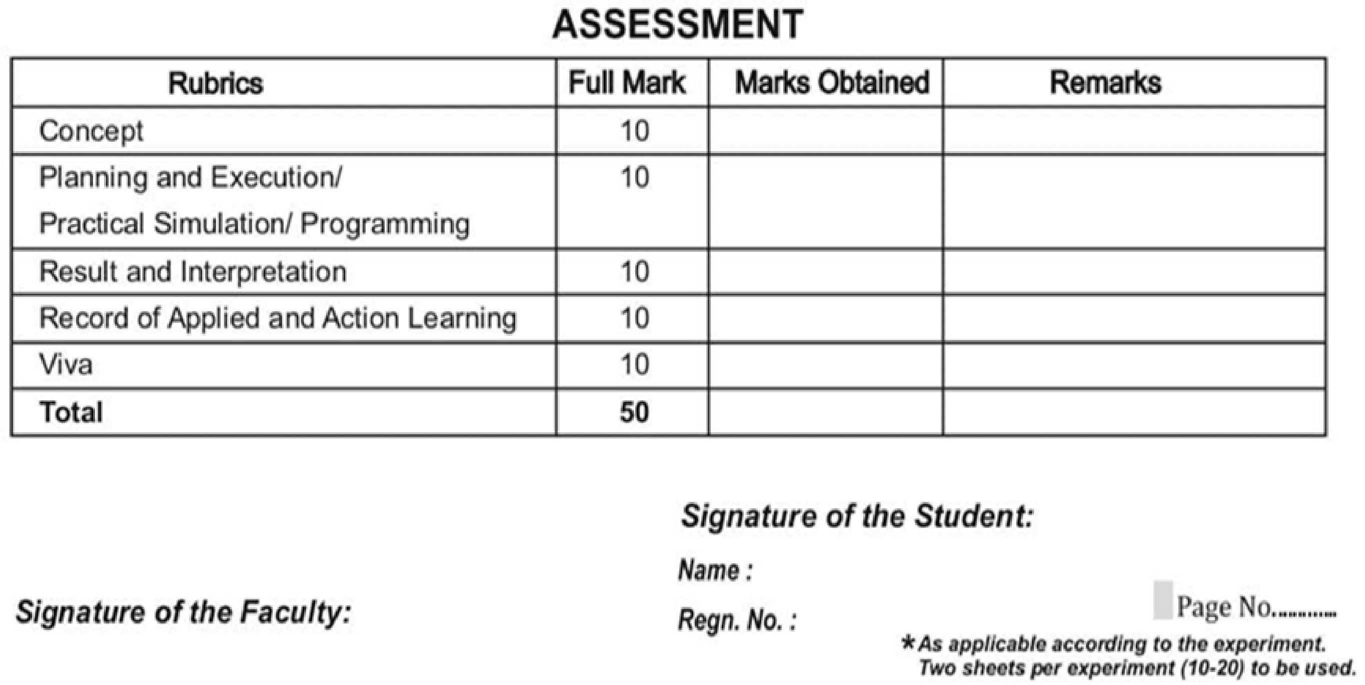
- Call contract functions using Web3.js or Ethers.js.
- Observe gas usage and transaction receipts.

⍰ **Check EVM Execution**

- Verify the execution of bytecode and storage updates on EVM.
- Monitor logs using the Ethereum client console.

Observation:

The Ethereum client ran successfully, accounts were created, and a smart contract was deployed. The EVM executed the contract, updated values, and showed gas usage and logs.



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.*