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 : Multi-Chain Deploy – BSC or Layer 2 Experience

Objective/Aim:

Deploy a simple smart contract on a multi-chain environment such as **Binance Smart Chain (BSC Testnet)** or an **Ethereum Layer-2 network** (e.g., Polygon Mumbai, Arbitrum Testnet). Connect MetaMask, configure the testnet, deploy the contract, and record deployment details and observations.

Apparatus/Software Used:

a. **MetaMask Wallet**
b. **Brave Web Browser**
c. **Remix IDE – https://remix.ethereum.org**
d. **BSC Testnet / Polygon Mumbai Testnet / Arbitrum Testnet**

Theory/Concept:

Multi-Chain Deployment:
Many EVM-compatible chains (BSC, Polygon, Arbitrum, Optimism) allow deploying the same Solidity smart contract with minimal changes. The deployment process remains identical to Ethereum due to EVM compatibility.

BSC (Binance Smart Chain):
A high-throughput chain using a modified Proof of Staked Authority consensus. Works with the same tools used for Ethereum.

Layer-2 Networks (Polygon, Arbitrum):
Scaling solutions built on top of Ethereum, enabling cheaper and faster transactions while retaining Ethereum security.

Procedure:

1. Open MetaMask and switch to the **BSC Testnet** or selected Layer-2 network.
2. Visit **Remix IDE** in your browser.
3. Create a new Solidity file and paste a simple contract (e.g., "HelloWorld").
4. Click the **Solidity Compiler** tab and compile the contract.
5. Go to **Deploy & Run Transactions** and select **Injected Provider (MetaMask)**.
6. Ensure the selected network is BSC Testnet / Layer-2 testnet.
7. Click **Deploy** and approve the transaction in MetaMask.
8. Wait for the deployment confirmation and record the **contract address**.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract HelloWorld {
    string public message;

    constructor() {    infinite gas 316400 gas
        message = "Hello, World!";
    }

    function setMessage(string memory newMessage) public {    infinite gas
        message = newMessage;
    }
}
```

Observation

Observation No.	Finding
1	Contract compiles and deploys on the Sepolia testnet without errors.
2	Access-controlled functions correctly restrict unauthorized users.
3	All Ether-handling functions include require() checks before state changes.
4	No reentrancy patterns detected in withdrawal or external call functions.
5	State changes occur before external calls where applicable, reducing risk of reentrancy.
6	No overflow/underflow vulnerabilities identified in arithmetic operations.

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 :