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:

To deploy and test a Solidity smart contract on **multiple blockchain networks**, such as **Binance Smart Chain (BSC)** and **Ethereum Layer-2 networks** (e.g., Polygon, Arbitrum, Optimism), to understand cross-chain deployment environments.

Apparatus/Software Used:

- VS Code – for coding and configuration
- Hardhat / Remix IDE – for compiling and deploying contracts
- MetaMask – wallet to connect to different networks
- RPC URLs (for BSC Testnet / Polygon Testnet)
- Node.js & npm – for package management
- Infura / Alchemy / Ankr – RPC providers

Theory:

Blockchain ecosystems like **BSC** and **Layer-2 networks** aim to solve Ethereum's scalability and gas issues. They maintain **EVM compatibility**, meaning the same Solidity code can run on multiple chains with minor configuration changes.

Multi-Chain Concepts

- **EVM Compatibility:** Smart contracts written in Solidity can be reused across networks.
- **RPC URLs:** Define endpoints for different networks (e.g., Ethereum, BSC, Polygon).
- **Chain ID:** Identifies each network uniquely.
- **Gas Fees:** Differ between chains; L2s and BSC are cheaper than Ethereum mainnet.
- **Bridges:** Allow token and data transfer between chains.

Procedure:

Multi-Chain Deploy – Stacks & Arbitrum Environment Setup

Objective:

To set up and configure development environments for deploying smart contracts on **Stacks (Bitcoi Layer)** and **Arbitrum (Ethereum Layer 2)**, enabling multi-chain dApp testing and interoperability

Steps / Algorithm

- **Environment Setup – Stacks (Clarity):**
  1. **Install Clarinet:**
     Use `npm install -g @hirosystems/clarinet` to install the Stacks development tool.
  2. **Initialize Project:**
     Run `clarinet new project_name` to create a new Clarity smart contract environment.
  3. **Write Smart Contract:**
     Create `.clar` files in the `/contracts` folder.
  4. **Test Contracts:**
     Execute `clarinet test` to run unit tests locally.
  5. **Deploy to Testnet:**
     Use `clarinet integrate` or **Stacks Explorer** for contract deployment on Testnet.

- **Environment Setup – Arbitrum (Solidity):**
  1. **Install Hardhat:**
     Use `npm install --save-dev hardhat` to create a Solidity project for Arbitrum.
  2. **Configure Network:**
     Add Arbitrum RPC in `hardhat.config.js`:

```
networks: {
 arbitrumSepolia: {
   url: "https://sepolia-rollup.arbitrum.io/rpc",
   accounts: [PRIVATE_KEY]
  }
}
```

  3. **Compile Contracts:**
     Run `npx hardhat compile`.
  4. **Deploy Contract:**
     Use a deploy script like `npx hardhat run scripts/deploy.js --network arbitrumSepolia`.
  5. **Verify on Explorer:**
     Check the deployed contract on **Arbiscan Testnet**.

Softwares Used

1. **Clarinet** – for Stacks smart contract development and testing.
2. **Hardhat** – for Solidity contract compilation and Arbitrum deployment.
3. **MetaMask** – for connecting wallet and interacting with both networks.

Output / Learning

Successfully configured and deployed smart contracts on both **Stacks** and **Arbitrum** environments, understanding differences in **Clarity (non-Turing complete)** and **Solidity (EVM-compatible)** ecosystems.

Observation:

1. The contract compiled and deployed successfully on both BSC Testnet and Polygon Mumbai.
2. Deployment gas fees on Polygon and BSC were significantly lower than Ethereum testnets.
3. Transaction confirmations were faster on Layer-2 networks.
4. Same Solidity code worked without modification across chains (EVM-compatible).
5. Verified both deployments on BscScan and PolygonScan using contract addresses.
6. Interaction with functions like updateMessage() worked smoothly on both networks.
7. RPC URL and chain ID configuration ensured successful multi-chain connectivity.

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.