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 :

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

Algorithm:

1.Set up development environments • Install Stacks CLI and Clarinet for Stacks contract deployment. • Install Hardhat or Remix + MetaMask (Arbitrum Testnet) for EVM-based deployment.
2.Write Smart Contracts • Create a Clarity contract for Stacks (e.g., token or storage logic). • Create a Solidity contract for Arbitrum implementing similar logic.
3.Compile Contracts • Use Clarinet test and build commands for Stacks. • Use Hardhat compile or Remix compiler for Arbitrum.
4.Deploy on Testnets • Deploy the Clarity contract on Stacks Testnet. • Deploy the Solidity contract on Arbitrum Sepolia Testnet using MetaMask and Hardhat.
5.Verify Deployment • Confirm deployment transaction on Stacks Explorer and Arbitrum Explorer. • Test function execution (e.g., token minting, data update).
6.Cross-Chain Observation • Analyze performance difference, transaction fee, and confirmation time between Stacks and Arbitrum.

* Softwares used

1.Clarinet
2.Hardhat
3.MetaMask

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

* Implementation Phase: Final Output (no error)

Multi-Chain Deploy – Stacks & Arbitrum Environment Setup

Objective:

To set up and configure development environments for deploying smart contracts on **Stacks (Bitcoin 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**.

* Implementation Phase: Final Output (no error)

• **Cross-Chain Deployment & Configuration:**
Set up and configured smart contract environments on **Stacks (Bitcoin Layer)** and **Arbitrum (Ethereum Layer 2)**, enabling seamless multi-chain deployment and interoperability testing.

• **Network Integration & Optimization:**
Configured RPC connections, wallets, and deployment scripts in **Hardhat** and **Clarinet**, ensuring efficient cross-network communication and optimized transaction flow.

• **Smart Contract Compatibility & Validation:**
Adapted Solidity and Clarity contracts to their respective environments, validating performance and ensuring proper function execution across both ecosystems.

• **Performance Monitoring & Final Testing:**
Deployed and verified contracts on **Stacks Testnet** and **Arbitrum Sepolia**, analyzing transaction confirmation speed, gas efficiency, and contract stability for consistent cross-chain behavior.

* Observations

1.Observed smoother and faster transaction execution on Arbitrum due to Layer-2 scalability enhancements.
2.Verified strong security and clarity in contract logic during deployment on the Stacks blockchain.
3.Noted effective cross-chain interoperability and consistent contract behavior across both testnets.

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. :

Signature of the Faculty:

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