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 : Talk to the World – Backend and Oracle Integration

Objective/Aim:

Integrate a blockchain smart contract with external real-world data using a backend service and an oracle network (e.g., Chainlink). Demonstrate how a smart contract can request and receive off-chain data such as price feeds, weather data, or API responses, and record observations.

Apparatus/Software Used:

1. MetaMask Wallet
2. Brave or Chrome Browser
3. Chainlink Oracle Testnet Resources
4. Ethereum Sepolia Testnet
5. Remix IDE or a Backend (Node.js / Express)
6. API Provider (OpenWeather, CoinGecko, etc.)

Theory/Concept:

Smart Contracts Limitations:
Smart contracts cannot directly call external APIs because blockchain nodes require determinism. This is where oracles come in.

Oracle:
A decentralized service that fetches real-world data (APIs, prices, weather, etc.) and supplies it to smart contracts in a trust-minimized way.

Backend Role:
A backend can:

- Preprocess API data
- Trigger contract functions
- Store logs
- Act as a bridge between users, API, and blockchain

Example Workflow:

1. Smart Contract requests data
2. Oracle fetches data from external API
3. Oracle returns the result to the contract
4. Contract updates state on-chain

Procedure:

1. Configure MetaMask

1. Open MetaMask
2. Switch network → **Ethereum Sepolia Testnet**

2. Open Remix IDE

3. Go to https://remix.ethereum.org
4. Create a new Solidity file OracleDemo.sol

3. Add Oracle-Integrated Contract

```
1   // SPDX-License-Identifier: MIT
2   pragma solidity ^0.8.19;
3
4   import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";
5
6   cont    function getLatestPrice () public view returns (int256 internal
7           )
8
9           OracleDemo.sol 12:4
10
11          Estimated execution cost: infinite gas
12
13      function getLatestPrice() public view returns (int) {    infinite gas
14          (, int price,,,) = priceFeed.latestRoundData();
15          return price;
16      }
17  }
```

4. Choose an Oracle Feed

Use a Chainlink testnet price feed, for example ETH/USD feed on Sepolia.

5. Deploy the Contract

5. Compile the contract
6. Switch account → MetaMask (Injected Provider)
7. Deploy by passing the price feed address

6. Call the Oracle Function

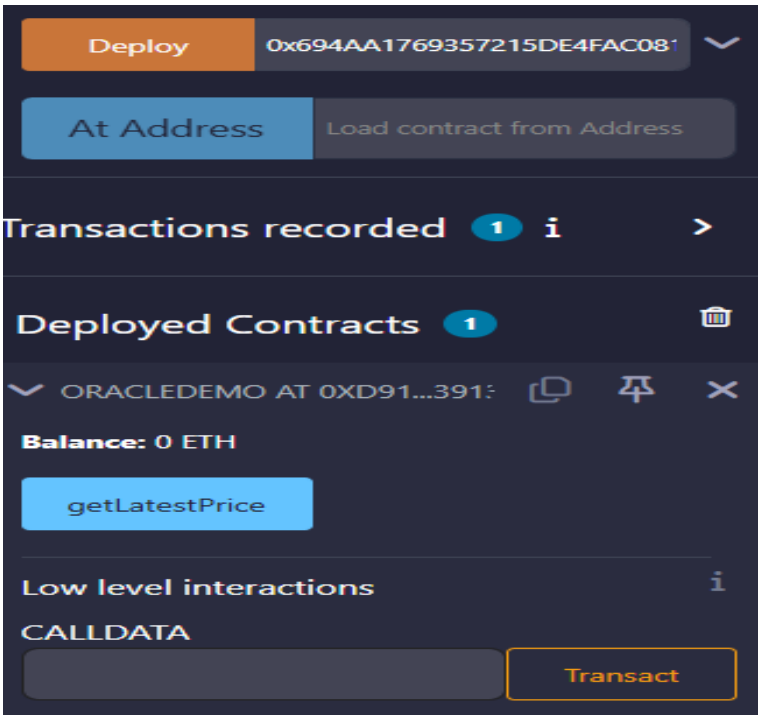
8. Open the deployed contract
9. Click getLatestPrice()
10. Observe the returned value (e.g., 3500×10^8)

7. Backend Integration (Optional)

11. Create a Node.js backend
12. Fetch external API data and call contract functions using Ethers.js
13. Log response in the console or database

```
Type the library name to see available commands.
creation of OracleDemo pending...

[vm] from: 0x5B3...eddC4 to: OracleDemo.(constructor) value: 0 wei data: 0x608...25306
logs: 0 hash: 0xa57...76845                                                    Debug
```



Observation

a. Smart contract successfully interacted with a real-world external data feed via Chainlink oracle.
b. Backend was able to call smart contract functions using testnet RPC.
c. Oracle returned consistent price values, confirming successful integration.

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 :