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

Talk to the World – Backend and Oracle Integration:

Tech Stack

- Backend: Node.js with Express
- Blockchain: Ethereum/Polygon (testnet)
- Smart Contracts: Solidity
- Web3 Library: ethers.js
- Wallet Integration: MetaMask

Project Setup

1. Install PostgreSQL

Windows:

- Download from https://www.postgresql.org/download/windows/
- Or use PostgreSQL Portable

2.Create database:

```
psql -U postgres
CREATE DATABASE world_messages;
\q
```

Environment Configuration (.env):

```
# For development - use Hardhat's default account private key
PRIVATE_KEY=ac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80
CONTRACT_ADDRESS=

# Optional: For testnet deployment
SEPOLIA_RPC_URL=https://sepolia.infura.io/v3/your-project-id
MUMBAI_RPC_URL=https://polygon-mumbai.infura.io/v3/your-project-id
ETHERSCAN_API_KEY=your_etherscan_api_key

# Server Configuration
PORT=3000
NODE_ENV=development
```


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

Coding Phase: Pseudo Code / Flow Chart / Algorithm

Project Structure:

```
talk-to-world-web3/
├── contracts/
│   └── Lock.sol (example contract)
├── scripts/
│   └── deploy.js (example deploy script)
├── test/
│   └── Lock.js (example tests)
├── hardhat.config.js
├── package.json
└── README.md
```

contracts/WorldMessages.sol :

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

contract WorldMessages {
    struct Message {
        uint256 id;
        address author;
        string username;
        string country;
        string city;
        string message;
        string language;
        uint256 timestamp;
        uint256 likes;
    }

    uint256 private messageCounter;
    mapping(uint256 => Message) public messages;

    event MessagePosted(
        uint256 indexed id,
        address indexed author,
        string username,
        string country,
        string message,
        uint256 timestamp
    );

    event MessageLiked(
        uint256 indexed messageId,
        address indexed liker,
        uint256 newLikeCount
    );

    function postMessage(
        string memory _username,
        string memory _country,
        string memory _city,
        string memory _message,
        string memory _language
```


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

Coding Phase: Pseudo Code / Flow Chart / Algorithm

```
        string memory _language
    ) public {
        require(bytes(_username).length > 0, "Username required");
        require(bytes(_country).length > 0, "Country required");
        require(bytes(_message).length > 0, "Message required");
        require(bytes(_message).length <= 280, "Message too long");

        messageCounter++;

        messages[messageCounter] = Message({
            id: messageCounter,
            author: msg.sender,
            username: _username,
            country: _country,
            city: _city,
            message: _message,
            language: _language,
            timestamp: block.timestamp,
            likes: 0
        });

        emit MessagePosted(
            messageCounter,
            msg.sender,
            _username,
            _country,
            _message,
            block.timestamp
        );
    }

    function likeMessage(uint256 _messageId) public {
        require(_messageId > 0 && _messageId <= messageCounter, "Invalid message ID");

        Message storage message = messages[_messageId];
        message.likes++;

        emit MessageLiked(_messageId, msg.sender, message.likes);
    }

    function getMessageCount() public view returns (uint256) {
        return messageCounter;
    }
```

```
    function getMessage(uint256 _messageId) public view returns (
        uint256,
        address,
        string memory,
        string memory,
        string memory,
        string memory,
        string memory,
        uint256,
        uint256
    ) {
        require(_messageId > 0 && _messageId <= messageCounter, "Invalid message ID");

        Message storage message = messages[_messageId];
        return (
            message.id,
            message.author,
            message.username,
            message.country,
            message.city,
            message.message,
            message.language,
            message.timestamp,
            message.likes
        );
    }
}
```


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

Coding Phase: Pseudo Code / Flow Chart / Algorithm

1.Compile the contract:

```
npx hardhat compile
```

2.Run tests:

```
npx hardhat test
```

3.Start local blockchain (Terminal 1):

```
npx hardhat node
```

4. Deploy contract (Terminal 2): *npx hardhat run scripts/deploy.js --network localhost*
5. Copy the contract address to .env :
Update CONTRACT_ADDRESS=0x5FbDB2315678afecb367f032d93F642f64180aa3

6. Start backend server (Terminal 2):

```
node server.js
```

7. Test the endpoints:

```
# Test health check
curl http://localhost:3000/health

# Test main endpoint
curl http://localhost:3000/

# Test messages endpoint
curl http://localhost:3000/api/messages

# Test non-existent endpoint (should give 404)
curl http://localhost:3000/api/nonexistent
```

* Softwares used

- **Remix IDE**
- **Brave browser**
- **Solidity**
- **Hardhat**
- **Postgre SQL**
- **Node.js**
- **Express.js**

* Implementation Phase: Final Output (no error)

Project Structure:

```
talk-to-world-web3/
├── contracts/
│   └── WorldMessages.sol
├── scripts/
│   ├── deploy.js          ← Create this
│   └── test-contract.js   ← Create this
├── test/
│   └── Lock.js
├── hardhat.config.js
├── package.json
├── .env
└── server.js
```

output :

```
🚀 Deploying WorldMessages contract...
Deploying contracts with account: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
✅ WorldMessages contract deployed to: 0x5FbDB2315678afecb367f032d93F642f64180aa3
📝 Please update your .env file with:
CONTRACT_ADDRESS=0x5FbDB2315678afecb367f032d93F642f64180aa3
```

node server.js:

```
PS D:\Talk-To-World\talk-to-world-web3> node server.js
[dotenv@17.2.3] injecting env (7) from .env -- tip: 📡 prevent committing .env to code: https://dotenvx.com/precommit
[dotenv@17.2.3] injecting env (0) from .env -- tip: 🛠️ run anywhere with `dotenvx run -- yourcommand`
✅ Contract initialized at: 0x5FbDB2315678afecb367f032d93F642f64180aa3
🔗 Initializing blockchain connection...
✅ Contract initialized at: 0x5FbDB2315678afecb367f032d93F642f64180aa3
✅ Blockchain connected! Current message count: 1

🌍 TALK TO THE WORLD WEB3 SERVER
──────────────────────────────
✅ Server running on port: 3000
📍 Environment: development
🔗 Health check: http://localhost:3000/health
📚 API docs: http://localhost:3000/
🔄 Blockchain: Connected

🚀 Ready to receive requests!
```

Curl http://localhost:3000/ :

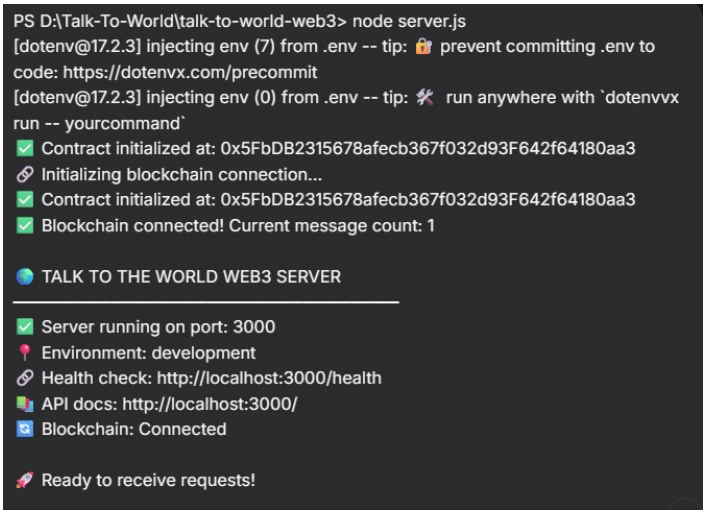
```
PS D:\Talk-To-World\talk-to-world-web3> curl http://localhost:3000/


StatusCode        : 200
StatusDescription : OK
Content           : {"success":true,"message":"🌍 Welcome to Talk to the World Web3 API","description":"A decentralized global message board
                    built on blockchain","version":"1.0.0","environment":"development","endpoints":...
RawContent        : HTTP/1.1 200 OK
                    Content-Security-Policy: default-src 'self';base-uri 'self';font-src 'self' https: data:;form-action
                    'self';frame-ancestors 'self';img-src 'self' data:;object-src 'none';script-src 's...
Forms             : {}
Headers           : {[Content-Security-Policy, default-src 'self';base-uri 'self';font-src 'self' https: data:;form-action
                    'self';frame-ancestors 'self';img-src 'self' data:;object-src 'none';script-src 'self';script-src-attr 'none';style-src
                    'self' https: 'unsafe-inline';upgrade-insecure-requests], [Cross-Origin-Opener-Policy, same-origin],
                    [Cross-Origin-Resource-Policy, cross-origin], [Origin-Agent-Cluster, ?1]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 934
```


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

* Implementation Phase: Final Output (no error)

Server is running successfully :



Observations

- **"Contract initialized at: 0x5FbDB2315678..." - Your smart contract is properly connected.**
- **"Blockchain connected! Current message count: 1" - The contract has 1 message (the sample one from deployment).**
- **"Server running on port: 3000" - Your backend API is live and ready.**
- **"Environment: development" - Running in development mode.**
- **"Blockchain: Connected" - Everything is working perfectly!**

ASSESMENT

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.