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

"Debugging Deep – Using Hardhat Console & Logs" : Happens throughout the development process. This is a crucial part of blockchain development.

1. Hardhat Console - Interactive Debugging

Accessing the Console:

```
# Connect to your local blockchain
npx hardhat console --network localhost
```

Run the debug script:

```
npx hardhat run scripts/debug-contract.js --network localhost
```

2.Common Debugging Scenarios:

Scenario 1: Transaction Reverts

```
# In Hardhat console
const tx = await contract.postMessage("", "Country", "Message", "English");
# Error: "Username required" - caught by require statement
```

Scenario 2: Gas Issues

```
# Check gas usage
const receipt = await tx.wait();
console.log("Gas used:", receipt.gasUsed.toString());
```

Scenario 3: Event Listening

```
// Listen for specific events in real-time
contract.on("MessagePosted", (id, author, username) => {
    console.log(`New message from ${username} (ID: ${id})`);
});
```


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

Coding Phase: Pseudo Code / Flow Chart / Algorithm

* Softwares used

Debugging Tools Summary:

1. Hardhat Console - Interactive debugging
2. Custom Debug Events - Track contract execution
3. Enhanced Logging - Detailed backend logs
4. Debug Scripts - Automated state checking
5. Debug API Endpoints - Real-time monitoring
6. Gas Analysis - Performance optimization

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

1. Smart Contract Debug Events Output

```
$ npx hardhat node

Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

🔍 [CONTRACT DEBUG] {
  message: 'postMessage called by',
  value: '0',
  account: '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266'
}
🔍 [CONTRACT DEBUG] {
  message: 'Input validation passed',
  data: 'ConsoleUser'
}
🔍 [CONTRACT DEBUG] {
  message: 'Message created with ID',
  value: '4',
  account: '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266'
}
📨 [CONTRACT EVENT] MessagePosted: {
  id: '4',
  author: '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266',
  username: 'ConsoleUser',
  country: 'TestLand',
  timestamp: '2024-01-15T10:30:45.000Z'
}
🔍 [CONTRACT DEBUG] {
  message: 'postMessage completed',
  value: '4',
  account: '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266'
}
```

2.Backend Debug Logging Output

```
$ node server.js

🔧 [DEBUG] Initializing BlockchainService...
✅ [DEBUG] Provider connected
✅ [DEBUG] Wallet initialized: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
📝 [DEBUG] Contract address: 0x5FbDB2315678afecb367f032d93F642f64180aa3
✅ Contract initialized at: 0x5FbDB2315678afecb367f032d93F642f64180aa3

📤 [DEBUG] Posting message to blockchain: {
  username: 'JohnDoe',
  country: 'Canada',
  messageLength: 24
}
⏳ [DEBUG] Transaction sent: 0x89a7c9b1c0e5f3a2d4b6e8f7c0a9b3d2e1f4a8c7b6e5d4a3b2c1e0f9a8b7c6d5
✅ [DEBUG] Transaction confirmed: {
  hash: '0x89a7c9b1c0e5f3a2d4b6e8f7c0a9b3d2e1f4a8c7b6e5d4a3b2c1e0f9a8b7c6d5',
  blockNumber: 9,
  gasUsed: "85432"
}
```

This "Debugging Deep" approach ensures you can quickly identify and fix issues throughout your blockchain development workflow :

1. Hardhat Console Output

```
$ npx hardhat console --network localhost
Welcome to Node.js v18.0.0.
Type ".help" for more information.
>
```

2.*Scenario: Out of Gas*

```
⏳ [DEBUG] Transaction sent: 0x...
❌ [DEBUG] Transaction failed: {
  error: 'out of gas',
  code: 'OUT_OF_GAS',
  gasLimit: '1000000',
  gasUsed: '1000000'
}
```

* Observations

Summary of Debug Outputs:

1. Contract State: Messages, balances, addresses
2. Transaction Details: Hashes, gas used, block numbers
3. Function Flow: Step-by-step execution tracking
4. Error Details: Exact revert reasons and locations
5. Event History: All blockchain events with parameters
6. Performance Metrics: Gas usage, timing information
7. Real-time Updates: Live event streaming

This comprehensive debugging output gives you complete visibility into your blockchain application's behavior at every level .

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.