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

Gas Race – Optimizing Smart Contract Efficiency :

Objective :

To compare gas usage of different versions of a smart contract and find the most gas-efficient implementation.

Procedure

Step 1: Choose a Simple Function or Contract

Pick a basic smart contract (for example, a Token contract, Storage contract, or Array handling contract).

```solidity
function store(uint _value) public {
    value = _value;
}
```

Step 2: Write Multiple Versions

Create different versions of the same function with small variations that may affect gas consumption, such as:

- Using storage vs memory
- Using uint8 vs uint256
- Using for loops vs while loops
- Using events vs no events
- Using mapping instead of array

```solidity
// Version A
function addToArray(uint num) public {
    arr.push(num);
}

// Version B (Optimized)
function addToArrayOptimized(uint num) public {
    uint len = arr.length;
    arr[len] = num;
}
```


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

Coding Phase: Pseudo Code / Flow Chart / Algorithm

Step 3: Deploy and Test
Use Hardhat or Remix IDE to deploy all versions on a testnet (like Sepolia) or local Hardhat network.
Then, call the same function multiple times for each version.

Step 4: Measure Gas Usage
Check the gas consumed for each transaction in:

- Remix IDE → "Details" → "Gas Used"
- Hardhat → use console.log or gasReporter plugin
- Etherscan → if deployed on testnet

Example (using Hardhat gas reporter):

```
npm install hardhat-gas-reporter
```

In hardhat.config.js:

```
require("hardhat-gas-reporter");
module.exports = {
  gasReporter: {
    enabled: true,
    currency: 'USD',
  },
};
```

Step 5: Compare Results
Record and compare the gas costs for each version.

Version	Function Change	Gas Used
A	Uses storage push	45,000
B	Pre-calculates index	28000

* Softwares used

- Solidity – for writing smart contracts
- Hardhat – for development, testing, and deployment
- Ethers.js – for blockchain interaction
- JavaScript / TypeScript – for scripting and testing
- Hardhat Gas Reporter – for measuring and comparing gas usage
- MetaMask – for connecting wallet and deploying on testnets
- Infura / Alchemy – for blockchain node access (RPC endpoint)
- Sepolia Testnet – for testing contract deployment and gas efficiency
- Remix IDE (optional) – for quick contract testing without full setup

Implementation Phase: Final Output (no error)

Initialize Hardhat Project:

```
bash

mkdir gas-race
cd gas-race
npm init -y
npm install --save-dev hardhat
npx hardhat
```

Install Additional Tools

```
npm install --save-dev hardhat-gas-reporter ethers chai mocha
```

Configure hardhat.config.js

```
require("hardhat-gas-reporter");

module.exports = {
  solidity: "0.8.24",
  gasReporter: {
    enabled: true,
    currency: "USD",
  },
};
```

Write Smart Contracts

- contracts/StorageA.sol
- contracts/StorageB.sol
  (Each with different logic)

Deploy and Test

- Run unit tests or deployment scripts.
- Observe gas used for each function.

```
npx hardhat test
```

* Implementation Phase: Final Output (no error)

Analyze Gas Reporter Output :

```
sql

·-----------------------------------------|----
|  Contract  |  Method          |  Gas Used |
|------------|------------------|-----------|
|  StorageA  |  store()         |  45000    |
|  StorageB  |  storeOpt()      |  23000    |
```

* Observations

Ideal Output / Deliverable

- **Report or Dashboard showing:**
  - **Contract versions tested**
  - **Gas used for each function**
  - **Comparison chart (if UI built)**
  - **Summary of which optimizations worked best**

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.