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

1. Start the process.
2. Install Node.js and npm on your computer
3. Set up Truffle by running npm install -g truffle
4. Create a new Truffle project using truffle init
5. Compile and deploy a sample smart contract with truffle compile and truffle migrate
6. Install Hardhat in your project using npm install --save-dev hardhat
7. Set up a Hardhat project by running npx hardhat
8. Compile and deploy the same smart contract with npx hardhat compile and npx hardhat run scripts/deploy.js
9. Save the results and take screenshots of each step
10. Finish the process.

* Softwares used

- npm (Node Package Manager): A tool used to install and manage JavaScript libraries and project dependencies.
- Truffle Suite: A framework for building, testing, and deploying Ethereum smart contracts.
- Hardhat: A flexible development environment for writing, compiling, and debugging Ethereum smart contracts.
- VS Code: A powerful text and code editor used by developers for writing and managing code.
- Ganache: A local blockchain simulator used to test and deploy smart contracts safely before going live.

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

Coding Phase: Pseudo Code / Flow Chart / Algorithm

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

* Testing Phase: Compilation of Code (error detection)

```
C:\Users\HP>npm install -g truffle
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache i
f you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerfu
l.
npm warn deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer supported
npm warn deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports prom
ises now, please switch to that.
npm warn deprecated har-validator@5.1.5: this library is no longer supported
npm warn deprecated yaeti@0.0.6: Package no longer supported. Contact Support at https://www.npmjs.com/support for more
info.
npm warn deprecated memdown@1.4.1: Superseded by memory-level (https://github.com/Level/community#faq)
npm warn deprecated glob@7.2.0: Glob versions prior to v9 are no longer supported
npm warn deprecated level-errors@2.0.1: Superseded by abstract-level (https://github.com/Level/community#faq)
npm warn deprecated encoding-down@6.3.0: Superseded by abstract-level (https://github.com/Level/community#faq)
npm warn deprecated deferred-leveldown@5.3.0: Superseded by abstract-level (https://github.com/Level/community#faq)
npm warn deprecated levelup@4.4.0: Superseded by abstract-level (https://github.com/Level/community#faq)
npm warn deprecated level-js@5.0.2: Superseded by browser-level (https://github.com/Level/community#faq)
npm warn deprecated level-packager@5.1.1: Superseded by abstract-level (https://github.com/Level/community#faq)
npm warn deprecated level-codec@9.0.2: Superseded by level-transcoder (https://github.com/Level/community#faq)
npm warn deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm warn deprecated multibase@0.6.1: This module has been superseded by the multiformats module
npm warn deprecated apollo-server-errors@3.3.1: The 'apollo-server-errors' package is part of Apollo Server v2 and v3, w
hich are now end-of-life (as of October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is
now found in the 'apollo/server' package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for m
ore details.
```

```
C:\Users\HP>npm install -g ganache-cli
npm warn deprecated ganache-cli@6.12.2: ganache-cli is now ganache; visit https://trfl.io/g7 for details

added 1 package in 6s

2 packages are looking for funding
  run `npm fund` for details
```

Install Hardhat

```
C:\Users\HP>cd hardhat-project

C:\Users\HP\hardhat-project>npm init -y
Wrote to C:\Users\HP\hardhat-project\package.json:

{
  "name": "hardhat-project",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```


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

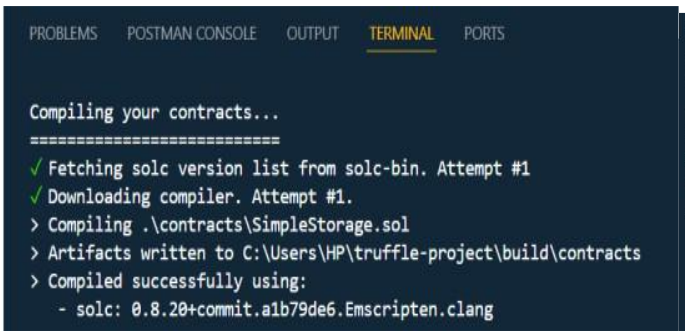
* **Implementation Phase: Final Output (no error)**

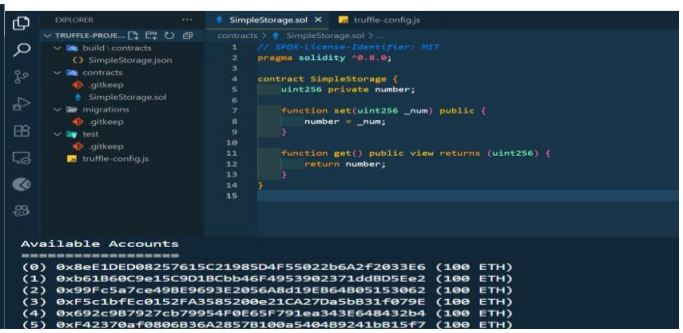
* Truffle project deployment process
* Install Ganache CLI (for local blockchain)
* npm install -g ganache-cli
* Create a new Truffle project
* mkdir truffle-project
* cd truffle-project
* truffle init
* Write smart contract (contracts/SimpleStorage.sol)
* Configure network in truffle-config.js
* Start local blockchain
* npx ganache-cli
* Compile the contract
* truffle compile
* Deploy (migrate) the contract
* truffle migrate --network development
* Open Truffle console for check deploy successfully

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

Implementation Phase: Final Output (no error)

Steps to Deploy Smart Contract in Hardhat :
1.Create a new folder for project
mkdir hardhat-project
cd hardhat-project
2.Initialize npm
npm init -y
4.Install Hardhat
npm install --save-dev hardhat
Setup Hardhat project
5.npx hardhat
Select “Create a JavaScript project”, press Enter for defaults.
6.Write smart contract (contracts/SimpleStorage.sol)
7.Add deployment script (scripts/deploy.js)
8.Compile the contract
9.npx hardhat compile
10.Start local Hardhat blockchain






```
async function main() {
  const SimpleStorage = await ethers.getContractFactory("SimpleStorage");
  const simpleStorage = await SimpleStorage.deploy();
  await simpleStorage.deployed();
  console.log("SimpleStorage deployed at:", simpleStorage.address);

  // Interact with contract
  await simpleStorage.set(42);
  const value = await simpleStorage.get();
  console.log("Stored value:", value.toString());
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});

```


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

* Implementation Phase: Final Output (no error)

```
C:\Users\HP\hardhat-project>> Block gas limit: 6721975 (0x6691b7)
'gas' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\HP\hardhat-project>
C:\Users\HP\hardhat-project>
C:\Users\HP\hardhat-project>1_initial_migration.js
'1_initial_migration.js' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\HP\hardhat-project>=====================
C:\Users\HP\hardhat-project>   Deploying 'Migrations'
'Deploying' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\HP\hardhat-project>   ----------------------
'----------------------' is not recognized as an internal or external command,
operable program or batch file.
```

* Observations

* Truffle is easy to use and good for beginners because it includes migration scripts for deployment, but it compiles slowly and has fewer debugging options.
  Hardhat, on the other hand, compiles faster, provides detailed error messages, and includes powerful debugging tools, making it ideal for advanced and production-level development.
  Both tools are used to compile, deploy, and test smart contracts, but while Truffle is simpler for newcomers, Hardhat is more modern, efficient, and developer-friendly

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 :

Signature of the Faculty:

Regn. No. :

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