Practical 7

Aim: Create a blockchain app for loyalty points with Hyperledger Fabric Ethereum Virtual

Creating the smart contract project

Installing hardhart

```
mkdir smart-contract
cd smart-contract
npm i -D hardhat
```

Creating a hardhat project

```
npx hardhat init
```

Choose **TypeScript**

```
? What do you want to do? …
  Create a JavaScript project
▶ Create a TypeScript project
  Create a TypeScript project (with Viem)
  Create an empty hardhat.config.js
  Quit
```

Rest can be left default

```
✔ What do you want to do? · Create a TypeScript project
✔ Hardhat project root: · /home/hackerman/web3/prac7/smart-contract
✔ Do you want to add a .gitignore? (Y/n) · y
✔ Help us improve Hardhat with anonymous crash reports & basic usage data? (Y/n)
 · n
```

Contents after project initialization

```
ls
```

```
→ smart-contract ls
contracts          node_modules  package-lock.json  scripts  tsconfig.json
hardhat.config.ts  package.json  README.md          test
→ smart-contract
```

Writing the smart contract:

Navigate the contracts directory and **delete** the existing Contract **Lock.sol** and Create a **new** Solidity file **LoyaltyPoints.sol**

```
rm contracts/Lock.sol
touch contracts/LoyaltyPoints.sol
ls ./contracts
```

```
→ smart-contract rm contracts/Lock.sol
→ smart-contract touch contracts/LoyaltyPoints.sol
→ smart-contract ls ./contracts
LoyaltyPoints.sol
→ smart-contract
```

Open the project in an editor and append the below code in **LoyaltyPoints.sol**

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract LoyaltyPoints {

    // model a member
    struct Member {
        address memberAddress;
        string firstName;
        string lastName;
        string email;
        uint points;
        bool isRegistered;
    }

    // model a partner
    struct Partner {
        address partnerAddress;
        string name;
        bool isRegistered;
    }

    // model points transaction
    enum TransactionType { Earned, Redeemed }
    struct PointsTransaction {
        uint points;
```

```
        TransactionType transactionType;
        address memberAddress;
        address partnerAddress;
    }

    //members and partners on the network mapped with their address
    mapping(address => Member) public members;
    mapping(address => Partner) public partners;

    //public transactions and partners information
    Partner[] public partnersInfo;
    PointsTransaction[] public transactionsInfo;

    //register sender as member
    function registerMember (string memory _firstName, string memory
_lastName, string memory _email) public {
      //check msg.sender in existing members
      require(!members[msg.sender].isRegistered, "Account already registered
as Member");

      //check msg.sender in existing partners
      require(!partners[msg.sender].isRegistered, "Account already
registered as Partner");

      //add member account
      members[msg.sender] = Member(msg.sender, _firstName, _lastName,
_email, 0, true);
    }

    //register sender as partner
    function registerPartner (string memory _name) public {
      //check msg.sender in existing members
      require(!members[msg.sender].isRegistered, "Account already registered
as Member");

      //check msg.sender in existing partners
      require(!partners[msg.sender].isRegistered, "Account already
registered as Partner");

      //add partner account
      partners[msg.sender] = Partner(msg.sender, _name, true);

      //add partners info to be shared with members
      partnersInfo.push(Partner(msg.sender, _name, true));

    }

    //update member with points earned
```

```
    function earnPoints (uint _points, address _partnerAddress ) public {
      // only member can call
      require(members[msg.sender].isRegistered, "Sender not registered as
Member");

      // verify partner address
      require(partners[_partnerAddress].isRegistered, "Partner address not
found");

      // update member account
      members[msg.sender].points = members[msg.sender].points + _points;

      // add transction
      transactionsInfo.push(PointsTransaction({
        points: _points,
        transactionType: TransactionType.Earned,
        memberAddress: members[msg.sender].memberAddress,
        partnerAddress: _partnerAddress
      }));

    }

    //update member with points used
    function usePoints (uint _points, address _partnerAddress) public {
      // only member can call
      require(members[msg.sender].isRegistered, "Sender not registered as
Member");

      // verify partner address
      require(partners[_partnerAddress].isRegistered, "Partner address not
found");

      // verify enough points for member
      require(members[msg.sender].points >= _points, "Insufficient points");

      // update member account
      members[msg.sender].points = members[msg.sender].points - _points;

      // add transction
      transactionsInfo.push(PointsTransaction({
        points: _points,
        transactionType: TransactionType.Redeemed,
        memberAddress: members[msg.sender].memberAddress,
        partnerAddress: _partnerAddress
      }));
    }

    //get length of transactionsInfo array
```

```
    function transactionsInfoLength() public view returns(uint256) {
        return transactionsInfo.length;
    }

    //get length of partnersInfo array
    function partnersInfoLength() public view returns(uint256) {
        return partnersInfo.length;
    }

}
```

Compiling the smart contract

Save the file and run the following

```
npx hardhat compile
```

```
→ smart-contract npx hardhat compile
Downloading compiler 0.8.24
Generating typings for: 1 artifacts in dir: typechain-types for target: ethers-v6
Successfully generated 6 typings!
Compiled 1 Solidity file successfully (evm target: paris).
→ smart-contract
```

Deploying the smart contract
We will be deploying the smart contract to the local hardhat network.

First we need to write the script that will deploy the smart contract open **scripts/deploy.ts** and replace its content with below code

Code:

```
import { ethers } from "hardhat";

async function main() {
  const loyaltyPoints = await ethers.deployContract("LoyaltyPoints");
  await loyaltyPoints.waitForDeployment();
  console.log(`LoyaltyPoints deployed to ${loyaltyPoints.target}`);
}

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

To deploy the contract we need to make sure the hardhat local network is running,

open a new terminal in the same project and run the following code and keep it open

```
npx hardhat node
```

```
➜ smart-contract npx hardhat node
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

Accounts
========

WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.

Account #0: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266 (10000 ETH)
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80

Account #1: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8 (10000 ETH)
Private Key: 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d
```

The output gives us the JSON RPC server information and some accounts that we can use

Go back to the previous terminal and deploy the contract

```
npx hardhat run scripts/deploy.ts --network localhost
```

```
➜ smart-contract npx hardhat run scripts/deploy.ts --network localhost
LoyaltyPoints deployed to 0x5FbDB2315678afecb367f032d93F642f64180aa3
➜ smart-contract
```

The contract is successfully deployed. **Save the address for later use**

Making the web app

Clone the given repository

```
git clone https://github.com/IBM/loyalty-points-evm-fabric.git
```

```
➜ prac7 git clone https://github.com/IBM/loyalty-points-evm-fabric.git
Cloning into 'loyalty-points-evm-fabric'...
remote: Enumerating objects: 1497, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 1497 (delta 0), reused 1 (delta 0), pack-reused 1494
Receiving objects: 100% (1497/1497), 4.66 MiB | 9.02 MiB/s, done.
Resolving deltas: 100% (336/336), done.
```

Navigate into the webapp directory and install the dependencies

```
cd loyalty-points-evm-fabric/web-app
npm i
```

```
→ prac7 cd loyalty-points-evm-fabric/web-app
→ web-app git:(master) npm i
npm WARN old lockfile
npm WARN old lockfile The package-lock.json file was c
npm WARN old lockfile so supplemental metadata must be
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, pleas
npm WARN old lockfile
npm WARN deprecated har-validator@5.1.0: this library
npm WARN deprecated uuid@3.3.2: Please upgrade  to ver
m for details.
```

Editing the dapp.js config file
There are a series of edits in the **dapp.js** file

First lets update the **contract address** and **provider.** Contract address will be the **address** it was deployed for you and provider will the **JSON RPC** url we got when we started the **hardhat node**

```
var contractAddress = '0x5fbdb2315678afecb367f032d93f642f64180aa3';
var provider = "http://127.0.0.1:8545";
```

Next we need to add contract ABI: for this open the **smart contract** project and navigate to **artifacts/contracts/LoyaltyPoints.sol** there will be a json file named **LoyaltyPoints.json** copy it into your **web app** project directory

```
cd artifacts/contracts/LoyaltyPoints.sol
cp LoyaltyPoints.json
../../../../loyalty-points-evm-fabric/web-app
cd ../../../../loyalty-points-evm-fabric/web-app
ls
```

```
→ smart-contract cd artifacts/contracts/LoyaltyPoints.sol
→ LoyaltyPoints.sol cp LoyaltyPoints.json ../../../../loyalty-points-evm-fabric/web-app
→ LoyaltyPoints.sol cd ../../../../loyalty-points-evm-fabric/web-app
→ web-app git:(master) ✗ ls
app.js  dapp.js  LoyaltyPoints.json  node_modules  package.json  package-lock.json  public  validate.js
→ web-app git:(master) ✗
```

Changing the code for **loyaltyABI** variable. Remove **loyaltyABI** variable and

replace it with the below code

```
const fs = require("fs")
const loyaltyABI = JSON.parse(fs.readFileSync("LoyaltyPoints.json")).abi
```

New code will look like this

```
    var address = contractAddress;

    console.log("Got address: " + address)

    const fs = require("fs")
    const loyaltyABI = JSON.parse(fs.readFileSync("LoyaltyPoints.json")).abi;

    var LoyaltyContract = web3.eth.contract(loyaltyABI);
    myContract = LoyaltyContract.at(address);
    return myContract;
}
```

Running the app

```
npm start
```

```
→  web-app git:(master) ✗ npm start

> loyaly-points@0.0.1 start
> node app.js

app running on port: 8000
```

Open the browser and enter the url http://localhost:8000