4 . Deploy an asset-transfer app using block chain. Learn app development within a Hyperledger Fabric network.

Change : Due to configuration concern , perform this lab using Ethereum

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract AssetTransfer {

struct Asset {

uint256 id;

string name;

address owner;

}

uint256 private nextAssetId;

mapping(uint256 => Asset) public assets;

event AssetCreated(uint256 id, string name, address owner);

event AssetTransferred(uint256 id, address from, address to);

// Create a new asset

function createAsset(string memory \_name) public {

uint256 assetId = nextAssetId;

assets[assetId] = Asset(assetId, \_name, msg.sender);

emit AssetCreated(assetId, \_name, msg.sender);

nextAssetId++;

}

// Transfer an asset to a new owner

function transferAsset(uint256 \_assetId, address \_newOwner) public {

require(assets[\_assetId].owner == msg.sender, "You are not the owner");

address previousOwner = assets[\_assetId].owner;

assets[\_assetId].owner = \_newOwner;

emit AssetTransferred(\_assetId, previousOwner, \_newOwner);

}

// Get details of an asset

function getAsset(uint256 \_assetId) public view returns (uint256, string memory, address) {

Asset memory asset = assets[\_assetId];

return (asset.id, asset.name, asset.owner);

}

}

5. Use block chain to track fitness club rewards..

Change : Due to configuration concern , perform this lab using Ethereum

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract FitnessClubRewards {

struct Member {

uint256 rewards; // Total rewards earned

bool isMember; // Membership status

}

address public owner;

mapping(address => Member) public members;

event MemberRegistered(address member);

event RewardsEarned(address member, uint256 points);

event RewardsRedeemed(address member, uint256 points);

modifier onlyOwner() {

require(msg.sender == owner, "Only owner can perform this action");

\_;

}

modifier onlyMember() {

require(members[msg.sender].isMember, "You must be a registered member");

\_;

}

constructor() {

owner = msg.sender;

}

// Register a new member

function registerMember(address \_member) public onlyOwner {

require(!members[\_member].isMember, "Already a member");

members[\_member] = Member(0, true);

emit MemberRegistered(\_member);

}

// Earn rewards for fitness activities

function earnRewards(address \_member, uint256 \_points) public onlyOwner {

require(members[\_member].isMember, "Not a registered member");

members[\_member].rewards += \_points;

emit RewardsEarned(\_member, \_points);

}

// Redeem rewards

function redeemRewards(uint256 \_points) public onlyMember {

require(members[msg.sender].rewards >= \_points, "Insufficient rewards");

members[msg.sender].rewards -= \_points;

emit RewardsRedeemed(msg.sender, \_points);

}

// View reward balance

function viewRewards() public view onlyMember returns (uint256) {

return members[msg.sender].rewards;

}

}

6.Build a web app that uses Hyperledger Fabric to track and trace member rewards.

Change : Due to configuration concern , perform this lab using Ethereum

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract FitnessClubRewards {

struct Member {

uint256 rewards; // Total rewards earned

bool isMember; // Membership status

}

address public owner;

mapping(address => Member) public members;

event MemberRegistered(address indexed member);

event RewardsEarned(address indexed member, uint256 points);

event RewardsRedeemed(address indexed member, uint256 points);

modifier onlyOwner() {

require(msg.sender == owner, "Only owner can perform this action");

\_;

}

modifier onlyMember() {

require(members[msg.sender].isMember, "You must be a registered member");

\_;

}

constructor() {

owner = msg.sender;

}

// Register a new member

function registerMember(address \_member) public onlyOwner {

require(!members[\_member].isMember, "Already a member");

members[\_member] = Member(0, true);

emit MemberRegistered(\_member);

}

// Earn rewards for fitness activities

function earnRewards(address \_member, uint256 \_points) public onlyOwner {

require(members[\_member].isMember, "Not a registered member");

members[\_member].rewards += \_points;

emit RewardsEarned(\_member, \_points);

}

// Redeem rewards

function redeemRewards(uint256 \_points) public onlyMember {

require(members[msg.sender].rewards >= \_points, "Insufficient rewards");

members[msg.sender].rewards -= \_points;

emit RewardsRedeemed(msg.sender, \_points);

}

// View reward balance

function viewRewards(address \_member) public view returns (uint256) {

return members[\_member].rewards;

}

}