**Practical 8**

**Aim:** Setting up and deploying a supply chain DApp smart contract in the Solidity programming language using Remix IDE.

**Background Information:**

**1. What is a DApp?**

* A DApp, short for decentralized application, runs on a blockchain network like Ethereum.
* It's made up of smart contracts that automate processes without needing a central authority.

**2. Smart Contracts and Solidity:**

* Smart contracts are self-executing contracts with code stored on the blockchain.
* Solidity is the language used to write these contracts for Ethereum.

**3. Remix IDE:**

* Remix is a web-based Integrated Development Environment (IDE) that helps create, edit, compile, and deploy smart contracts on the Ethereum blockchain.

**4. Supply Chain DApp:**

* This DApp manages the flow of products within a supply chain using a smart contract.
* It tracks product creation, ownership transfers, and history.

**Installation steps:**

**Prerequisites:**

Remix IDE: Access the Remix Integrated Development Environment (IDE) here **(**<https://remix.ethereum.org/>).

Solidity Compiler: Ensure the Solidity compiler is available within Remix IDE.

To set up and deploy a Lottery Smart Contract using Solidity programming language in Remix IDE, follow these steps:

**Steps:**

1. **Open Remix IDE:**

Navigate to the [Remix IDE website] (https://remix.ethereum.org/) in your web browser.

1. **Create a New File:**

* Click on the "+" button on the left-hand side panel to create a new file.
* Name the file ‘SupplyChain.sol’ or any desired name for your contract.

1. **Write the Smart Contract Code:**

* Write the Solidity code for the Supply Chain Smart Contract in the created file.
* Ensure to define necessary variables, mappings, functions, and constructor (if needed).
* In the newly created file, write your Solidity code for the voting smart contract.

-Here's a basic example:

// SPDX-License-Identifier: GPL-3

pragma solidity ^0.8.0;

/\*\*

\* @title Storage

\* @dev Store & retrieve value in a variable

\* @custom:dev-run-script ./scripts/deploy\_with\_ethers.ts

\*/

contract SupplyChain {

uint public productCount = 0;

struct Product {

uint id;

string name;

uint quantity;

address owner;

address payable[] history;

}

mapping(uint => Product) public products;

event ProductCreated(uint id, string name, uint quantity, address owner);

event ProductTransferred(uint id, address from, address to);

function createProduct(string memory \_name, uint \_quantity) public {

productCount++;

address payable[] memory initialHistory;

products[productCount] = Product(productCount, \_name, \_quantity, msg.sender, initialHistory);

emit ProductCreated(productCount, \_name, \_quantity, msg.sender);

}

function transferProduct(uint \_productId, address \_newOwner) public {

require(\_productId > 0 && \_productId <= productCount, "Invalid ID");

Product storage product = products[\_productId];

require(msg.sender == product.owner, "Only the owner can transfer the product");

product.owner = \_newOwner;

product.history.push(payable(\_newOwner));

emit ProductTransferred(\_productId, msg.sender, \_newOwner);

}

function getProductHistory(uint \_productId) public view returns (address payable[] memory) {

require(\_productId > 0 && \_productId <= productCount, "Invalid product ID");

return products[\_productId].history;

}

}

1. **Compile the Contract:**

* Go to the "Solidity Compiler" tab in Remix IDE.
* Select the version of Solidity used in your contract (e.g., `0.8.0`).
* Click on the "Compile SupplyChain.sol" button. Ensure there are no compilation errors.

1. **Deploy & Run Transactions:**

**-**Move to the "Deploy & Run Transactions" tab in Remix IDE.

* Under the "Deploy" section:
* Select the contract you want to deploy from the dropdown list (e.g., `SupplyChain`).
* Adjust constructor arguments if your contract requires any.
* Click on the "Deploy" button.
* Click on Deployed Contract🡪 then click on SUPPLYCHAIN AT 0xd9145CCE52D386f254917e481eB44e9943F39138
* Then click on createProduct downward arrow: \_name: “string”, \_quantity: uint256 will appear
* Enter the product name (“Cake”) and quantity (50)🡪click on transact.
* Then in product write 1🡪then click on downward arrow🡪 click call button.

-Remix will display transaction details, logs, execution cost (gas usage), and transaction hash for each action executed.

1. **Generating UML Diagram:**

* Utilize UML (Unified Modelling Language) tools or software to create an architectural diagram.
* Include components like the smart contract, participants, manager, transactions.
* Illustrate the design and implementation of your supply chain DApp using UML diagrams for clarity and documentation purposes.
* For creating a UML diagram:

-Right click on ‘SupplyChain.sol’ file.

-then click on Generate UML.

**Conclusion:**

Thus, the setting up and deploying a supply chain DApp smart contract in the Solidity programming language using Remix IDE is performed successfully.