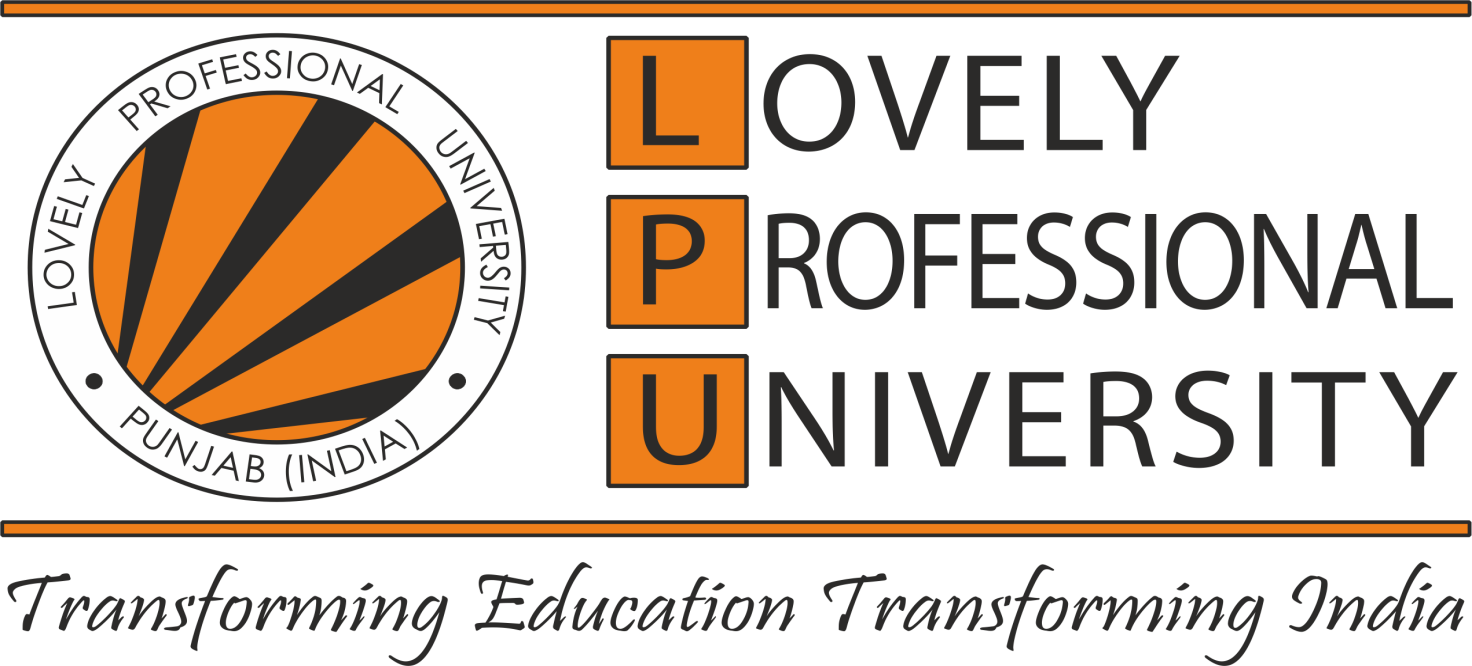
CSC307 – Blockchain Architecture and Design

### **By**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr. No. | Registration No | Name of Students | Roll No | Total Marks | Marks Obtained | Signature |
| 1 | 12202166 | Samuel Singh | 24 |  |  |  |



### 

**Submitted To: Mr. Piyush Gururani**

Lovely Professional University

Jalandhar, Punjab, India.

**Problem Statement**

4. Create a Solidity contract that tracks goods as they move through different stages in a supply chain. Define a function to allow an authorized entity to update the status of a shipment (e.g., “In Transit,” “Delivered”). Ensure that only authorized entities can update the status, and users can view the status. Include code for role-based access control.

**Solution:**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract SimpleSupplyChain {

// The owner of the contract (usually the deployer)

address public owner;

// Mapping to store shipment statuses using a unique shipment ID

mapping(uint256 => string) public shipmentStatus;

// Event emitted whenever a shipment status is updated

event StatusUpdated(uint256 indexed shipmentId, string newStatus);

// Modifier to allow only the owner to perform certain actions

modifier onlyOwner() {

require(msg.sender == owner, "Not authorized"); // Ensure the caller is the owner

\_;

}

// Constructor sets the deployer as the owner of the contract

constructor() {

owner = msg.sender; // Assign deployer address to the owner variable

}

// Function to update the status of a shipment

function updateStatus(uint256 shipmentId, string memory newStatus) public onlyOwner {

shipmentStatus[shipmentId] = newStatus; // Update the shipment status

emit StatusUpdated(shipmentId, newStatus); // Emit the status update event

}

// Function to view the current status of a shipment

function getStatus(uint256 shipmentId) public view returns (string memory) {

return shipmentStatus[shipmentId]; // Return the status of the given shipment ID

}

}

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Key Features in This Contract:

1. Custom Role-Based Access Control:
   * Admin: The deployer of the contract is set as the admin. Only the admin can manage updaters.
   * Updater: Admin can authorize addresses as updaters using addUpdater and revoke them using removeUpdater.
2. Shipment Tracking:
   * Shipments: Each shipment is stored in the shipments mapping. The status can only be updated by authorized updaters.
3. Events:
   * ShipmentStatusUpdated: Logs status changes with the shipment ID, new status, and the updater's address.
   * UpdaterRoleChanged: Tracks when an address is granted or revoked the updater role.
4. Utility Functions:
   * addUpdater: Admin can grant the updater role to an address.
   * removeUpdater: Admin can revoke the updater role from an address.
   * getShipment: Anyone can view shipment details, including ID, status, and the last updater's address.

Deployment and Usage:

1. Deploy the Contract:
   * The deployer becomes the admin by default.
2. Manage Updaters:
   * Admin can call addUpdater(address) to authorize an updater.
   * Admin can call removeUpdater(address) to revoke updater permissions.
3. Update Shipment Status:
   * Authorized updaters can call updateShipmentStatus(shipmentId, newStatus) to change the status of a shipment.
4. View Shipment Details:
   * Anyone can call getShipment(shipmentId) to retrieve shipment details.

This implementation ensures clarity, transparency, and proper role management while maintaining simplicity for real-world usage.

Functions and Purpose (Extremely Short)

1. addUpdater(address updater)
   * Admin adds an address as an updater.
2. removeUpdater(address updater)
   * Admin removes an updater.
3. updateShipmentStatus(uint256 shipmentId, string calldata newStatus)
   * Updater changes shipment status.
4. getShipment(uint256 shipmentId)
   * Retrieve shipment details.
5. constructor()
   * Sets the deployer as admin.

Modifiers

1. onlyAdmin()
   * Restricts access to admin.
2. onlyUpdater()
   * Restricts access to updaters.

Data Types

1. address
   * Represents Ethereum addresses (e.g., admin, updater, lastUpdatedBy).
2. bool
   * Stores true/false values (e.g., isUpdater[updater]).
3. uint256
   * Represents unsigned integers, used for shipment IDs (e.g., shipmentId).
4. string
   * Holds text data, such as shipment status (e.g., "In Transit", "Delivered").

Complex Types

1. struct Shipment
   * Custom type to store shipment details:
     + id (uint256): Shipment ID.
     + status (string): Current status of the shipment.
     + lastUpdatedBy (address): Address of the last updater.
2. mapping(uint256 => Shipment)
   * Maps shipment IDs to Shipment structs for tracking shipment details.
3. mapping(address => bool)
   * Maps addresses to a boolean indicating if they have the updater role (isUpdater).