Draw a use case diagram for Library Management System is an automation system used to manage a library and the different resource management required in it like cataloguing of books, allowing check out and return of books, invoicing, user management, etc. The user can search for books details using few book properties (Book ID, Title, Author, and Publisher). Searching should return details about all the book copies that match the search query.

## **Aim**

To design a **Library Management System** using **UML (Unified Modeling Language)** to automate and streamline library operations such as **book borrowing, returning, catalog management, and user tracking**.

## **Procedure (Step-by-Step Process)**

### **Step 1: Identify Key Entities (Classes) in the System**

The main components of a Library Management System include:

1. **User (Member/Librarian)** – Handles book borrowing and returning.
2. **Book** – Represents books in the library catalog.
3. **Library** – Manages books, users, and transactions.
4. **Transaction** – Records borrowing and returning of books.
5. **Fine** – Manages late return penalties.

\

### **Step 2: Define Class Attributes and Methods**

#### **1. User Class (Generalized as Member and Librarian)**

* **Attributes**: **userID, name, email, phoneNumber, borrowedBooks[]**
* **Methods: register(), borrowBook(), returnBook(), payFine(), viewCatalog()**

#### **2. Librarian Class (Inherits from User)**

* **Attributes**: librarianID, position
* **Methods**: addBook(), removeBook(), updateBookDetails(), manageUsers(), issueBook()

#### **3. Book Class**

* **Attributes**: bookID, title, author, ISBN, status (Available/Borrowed), borrowerID
* **Methods**: updateStatus(), getBookDetails()

#### **4. Library Class**

* **Attributes**: libraryName, location, books[], users[]
* **Methods**: searchBook(), addUser(), removeUser(), trackBookAvailability()

#### **5. Transaction Class**

* **Attributes**: transactionID, userID, bookID, issueDate, returnDate, status
* **Methods**: recordTransaction(), calculateDueDate(), checkOverdue()

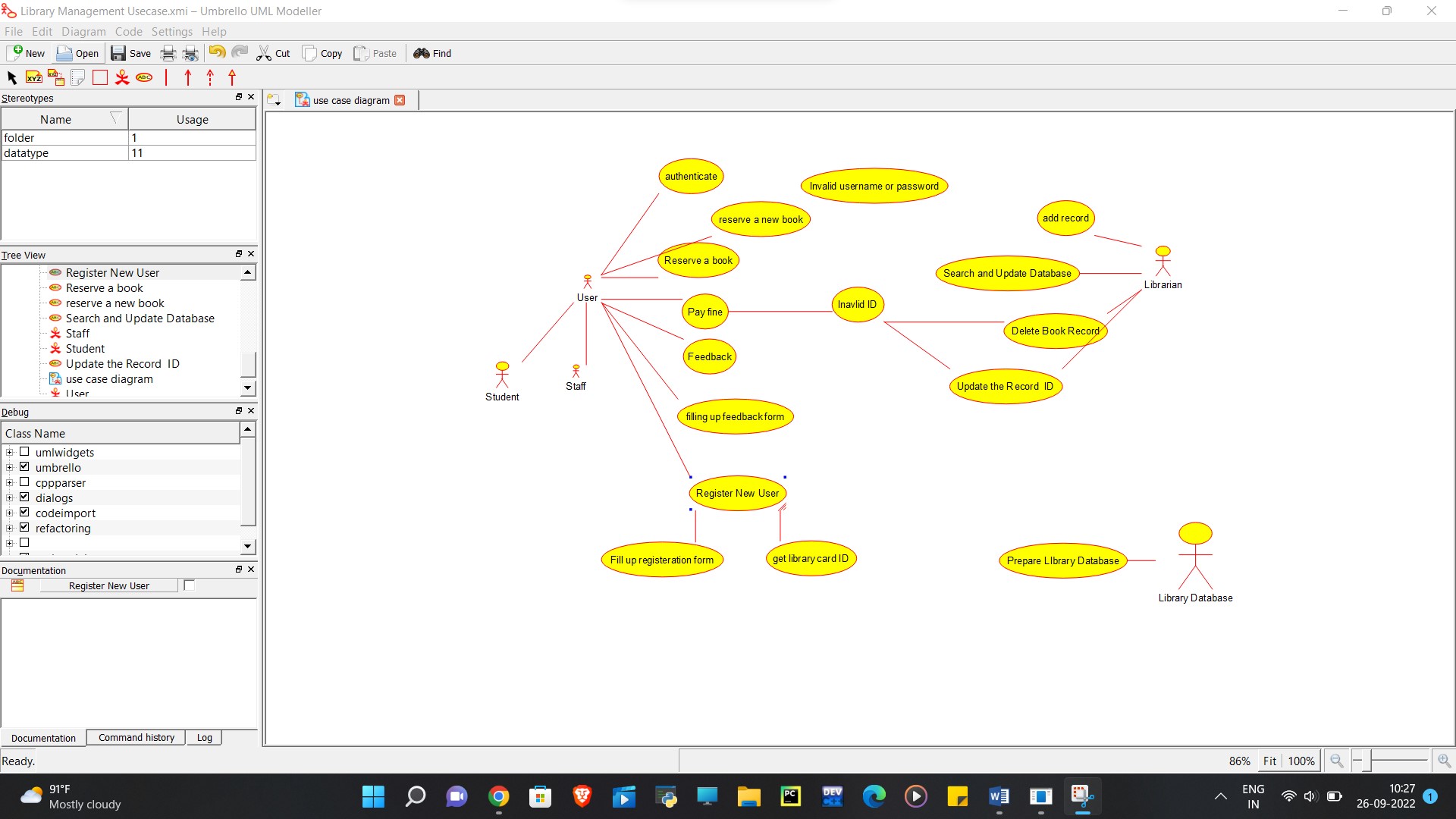
#### **6. Fine Class**

* **Attributes**: fineID, userID, amount, status (Paid/Unpaid)
* **Methods**: calculateFine(), updateFineStatus(), collectFine()

### **Step 3: Define Relationships Between Classes**

* **User** (Member) borrows or returns **Books**.
* **Librarian** manages **Books** and **Users**.
* **Library** maintains **Books**, **Users**, and **Transactions**.
* **Transaction** records book **borrowing and returning**.
* **Fine** is issued for **late returns**.

Usecase diagram



## **Result**

The **UML Class Diagram** for the **Library Management System** models the relationships and behaviors of library entities. It provides a structured approach for managing **book borrowing, returns, fines, and user accounts** while improving efficiency and organization.