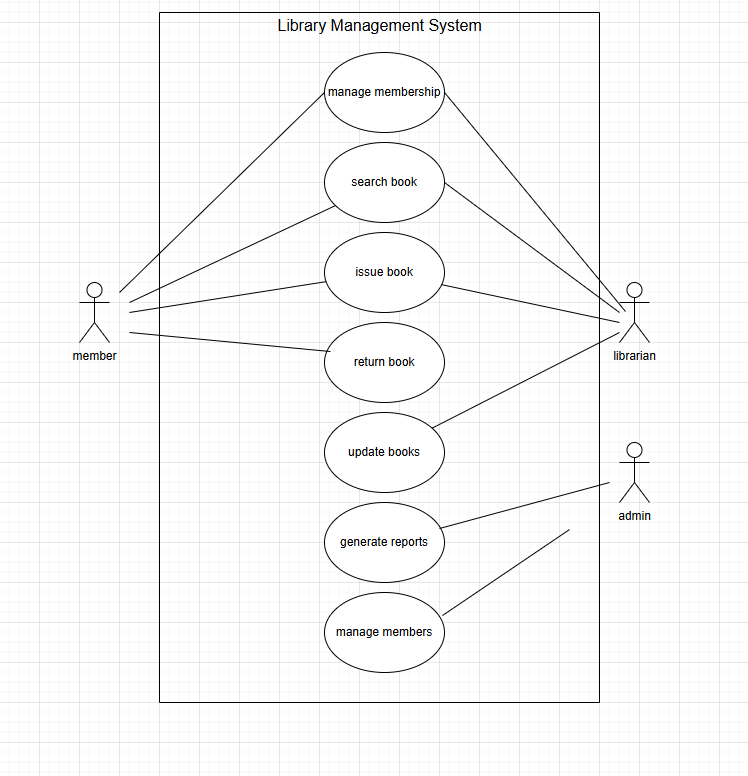
# Chapter 1: Use Case of Project – Library Management System



# Khadija zamir SP23-BSE-025

# Fully Dressed Use case: generate report

|  |  |
| --- | --- |
| **Section** | **Details** |
| **Use Case Name** | Generate Reports |
| **Primary Actor** | Admin |
| **Trigger** | Admin clicks on the "Generate Report" button |
| **Preconditions** | 1. Admin must be logged into system 2. Relevant data (e.g., borrowing records, returns, catalog) must exist |
| **Main Success Scenario** | 1. Admin selects the “Generate Reports” function  2. Specifies the type of report (e.g., borrowing frequency, overdue books, catalog additions)  3. Choosing report format (PDF, Excel)  4. System processes the data.  5. The system generates and displays the report.  6. Admin reviews and downloads it |
| **Postconditions** | 1. Report is generated and available for download/view 2. Report is stored for future reference |
| **Alternate Flows** | **2a:** If type or format is not specified, system prompts for missing info  **4a:** If there is no data, the system shows error "No data available for the selected criteria" |
| **Special Requirements** | 1. Report generation must be completed within 10 seconds.  2. Downloadable in PDF and Excel.  3. Stored with timestamp and admin ID.  4. Filter options: date range, department, or category |

# A diagram with text and arrows AI-generated content may be incorrect.System Sequence Diagram

# A screenshot of a computer AI-generated content may be incorrect.Class diagram:

# Coding Standards – Generate Reports

**The "Generate Reports" feature is implemented in Java using standard object-oriented practices. The following coding standards were followed:**

* **Class names use PascalCase (e.g., ReportGenerator)**
* **Method and variable names use camelCase (e.g., generateReport, reportType)**
* **Indentation is set to 4 spaces**
* **Braces are placed on the same line as the statement**
* **Comments use // for single-line and /\* \*/ for method or block explanations**
* **Each class is placed in a separate file, and methods are kept small and specific**
* **Error handling includes checking for missing input or data, using try-catch for exceptions**

# Nida khan SP23-BSE-011

**Use Case: Search Book**

# Fully Dressed Use case: Search Book

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case ID** | UC002 |
| **Use Case Name** | Search Book |
| **Brief Description** | This use case describes how a library user searches for a book using the Library Management System by entering criteria such as title, author, etc. |
| **Primary Actor** | Library User |
| **Secondary Actors** | Librarian, Library Management System |
| **Preconditions** | • User must have access to the LMS (via web, mobile, or terminal).• The catalog must be up-to-date and indexed. |
| **Postconditions** | • The user sees matching books or is notified that no matches are found.• The search is logged for analytics or audit. |
| **Trigger** | The user selects the "Search Book" option from the Library Management System interface. |
| **Basic Flow** | 1. User accesses the LMS.2. Navigates to the search page.3. Enters search criteria (e.g., title, author).4. System validates input.5. System queries the catalog.6. System displays matching results.7. User views the list. |
| **Alternative Flows** | **3a. Invalid Input Format:**– System displays an error if input (e.g., ISBN) is invalid.– User re-enters correct input.  **6a. No Matching Results:**– System notifies user of no matches.– User is prompted to refine search. |
| **Exception Flows** | **a) System Downtime:**– Search unavailable; message shown.  **b) Database Error:**– Error accessing catalog; log generated.**c) Authorization Issue:**– Restricted data access denied; message shown. |
| **Business Rules** | • ISBN must be 10 or 13 digits.• Search must return results in ≤ 5 seconds.• Restricted books are only visible to authorized users.• All search activities are logged with user ID.• Catalog must be indexed at least every 24 hours. |
| **Assumptions** | • The system supports partial and multi-field searches.• The search uses current catalog data.• High availability is ensured during peak usage. |

# System event design:

|  |  |  |
| --- | --- | --- |
| **Actor Action (Trigger)** | **System Event** | **System Description / Response** |
| **User selects "Search Book" option** | initiateSearch() | System loads the search page and prepares for input. |
| **User enters search criteria (title, author, etc.)** | — | Data entered manually — not a system event (user input). |
| **User clicks "Search"** | validateSearchInput(searchData) | System validates input, ensuring criteria are in the correct format (e.g., ISBN, title, etc.). |
| **System validates search input** | checkSearchCriteria() | System checks for the correct format (e.g., ISBN is 10 or 13 digits) and that all necessary fields are filled. |
| **System queries the catalog for matches** | queryCatalog(searchData) | System queries the database using the search criteria (e.g., title, author) to find matching books. |
| **System processes search results** | processSearchResults() | System processes the query results and prepares them for display, sorting results if needed. |
| **System displays search results** | displaySearchResults() | System displays the matching books or a message indicating no matches were found. |
| **User views results** | — | User can view the list of results or be prompted to refine search if no matches are found. |
| **System logs search activity** | logSearchActivity(userID, searchData) | System logs the search action with user ID and search criteria for analytics or audit. |

# System Sequence Diagram:

A black diagram with white text

AI-generated content may be incorrect.

# CLASS DIAGRAM:

A screenshot of a computer

AI-generated content may be incorrect.

# SEQUENCE DIAGRAM:

A black screen with white text

AI-generated content may be incorrect.

# Fully Dressed Use Case: Return Book

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Field** | **Details** | | **Use Case ID** | UC003 | | **Use Case Name** | Return Book | | **Brief Description** | This use case describes how a user (library member) returns a previously borrowed book using the Library Management System (LMS). The system verifies the return, updates records, and notifies the librarian if needed. | | **Primary Actor** | Library User | | **Secondary Actors** | Librarian, Library Management System | | **Preconditions** | • The user must be logged into the LMS.  • The book must be issued to the user in the system.  • The system must be online and connected to the book database. | | **Postconditions** | • The book status is updated to "Available".  • The user's account reflects the return.  • Any fines (if applicable) are calculated and shown. | | **Trigger** | The user selects the "Return Book" option from the LMS or physically returns the book to the librarian. | | **Basic Flow** | 1. User accesses the LMS and selects "Return Book".  2. Enters or scans book ID/ISBN.  3. System verifies the book is issued to the user.  4. System updates the status of the book as returned.  5. Fine (if any) is calculated based on due date.  6. System updates the user's account and history.  7. Confirmation is shown to the user. | | **Alternative Flows** | 3a. Book not issued to user:  – System shows an error message.  – User is asked to contact librarian.   5a. Overdue Return:  – System calculates fine.  – Fine is shown and added to user's dues. | | **Exception Flows** | a) System Offline:  – Return cannot be processed; manual logging advised.   b) Book Already Returned:  – System shows status and blocks duplicate return.   c) Invalid Book ID:  – System notifies the user and requests re-entry. | | **Business Rules** | • Returned books must update inventory instantly.  • Fine is ₹10 per day after the due date.  • Book can’t be returned if not issued to the user.  • All return activities are logged with timestamp and user ID. | | **Assumptions** | • Barcode scanner or manual entry is available.  • Fine payment can be settled later.  • Return can be initiated online or at the help desk | |

# System Sequence Diagram:

A black screen with white text

AI-generated content may be incorrect.

# Sequence Diagram:

A black screen with white text

AI-generated content may be incorrect.

# Class Diagram:

A screenshot of a computer

AI-generated content may be incorrect.

# GUL-E-NARJIS SP23-BSE-023

# Fully dressed USE CASE: UPDATE BOOK:

|  |  |
| --- | --- |
| Use Case ID: | UC001 |
| Use Case Name: | Update Book Information |
| Brief Description: | This use case describes the process for updating an existing book's information in the system. The book's details, such as title, author, genre, price, and availability status, can be modified by an authorized user (such as a librarian or admin). |
| Primary Actor**:** | Admin/Librarian |
| Secondary Actor**:** | Book System |
| Preconditions: | * The user must be authenticated as an authorized admin or librarian. * The book to be updated must already exist in the system. * The system must be operational, and the database must be accessible. |
| Postconditions: | * The book's details are updated in the system. * The system reflects the changes across all relevant views (book catalog, search results, etc.). * The updated book information is stored in the database. |
| Trigger: | * The user selects the "Update Book" option from the book management interface. |
| Basic Flow (Main Success Scenario): | 1. **User logs in** to the system as an authorized admin or librarian. 2. **User navigates** to the book management section. 3. **User selects** a book from the list of available books to update. 4. The system displays the book's current details (e.g., title, author, genre, price, availability). 5. **User edits** the desired fields (e.g., updates title, author, price, genre, or availability). 6. **User submits** the updated information. 7. **The system validates** the updated information (e.g., checks for required fields, ensures no duplicate book titles, validates price as a positive number). 8. **The system updates** the book information in the database. 9. The system displays a **success message** confirming that the book details have been successfully updated. 10. The system **reflects** the updated details across all relevant sections, including search results and book catalogs |
| Alternative Flows: | **User logs in to the system as an authorized admin or librarian.** **1a. (Invalid Data Input):**  If the user enters invalid login credentials (e.g., incorrect username or password), the system will:   1. Display an error message indicating the invalid credentials. 2. Allow the user to re-enter the correct login credentials.   **User navigates to the book management section.** **2a. (Unauthorized User):**  If the user does not have proper authentication (i.e., is not an authorized admin or librarian), the system will:   1. Deny access to the "Update Book" functionality. 2. Display an error message indicating insufficient permissions.   **The system displays the book's current details (e.g., title, author, genre, price, availability).** **4a. (Database Error):**  If the system encounters an error while displaying the book's details (e.g., due to corrupted data), the system will:   1. Display an error message indicating that the book details could not be retrieved. 2. Prompt the user to try again later or provide an option to proceed with a partial view. |
| Exception Flows: | * 1. **Exception Flow 1 (Book Not Found)**:   If the user tries to update a book that does not exist in the system, the system will:   1. Display a message indicating that the book was not found. 2. Allow the user to select another book or exit the update process.    1. **Exception Flow 2 (Network Failure)**:   If the system loses connection to the database or encounters a network issue:   1. Display an error message indicating the network issue. 2. Suggest the user retry the update once the connection is restored. |
| Business Rules: | 1. A book's title must be unique in the system. Two books cannot have the same title. 2. The price of a book must be a positive number. 3. A book must be marked as either "available" or "unavailable." |
| Assumptions: | 1. The system will handle concurrency issues (e.g., preventing two users from updating the same book simultaneously). 2. The system will maintain a backup of the original book information for auditing or recovery purposes. |

**System event design:**

|  |  |  |
| --- | --- | --- |
| **Actor Action (Trigger)** | **System Event** | **Response/Outcome** |
| Selects "Update Book" | updateBookInfo(bookID) | System retrieves book details and displays them for editing. |
| Enters login credentials | authenticateUser (username, pwd) | System verifies credentials and sets access level. |
| Edits book details | submitUpdatedInfo(data) | System validates fields (title uniqueness, positive price, availability). |
| Submits form | validateAndUpdateBook(data) | System writes validated info to the database and updates relevant UI views. |
| Book not found | retrieveBook(bookID) | If null, system shows "Book not found." |
| Validation fails | validateInput(data) | System prompts user with appropriate error messages. |
| System/database error | updateBookInDB(data) | System displays error and maintains old state. |

# Class diagram :

A diagram of a computer

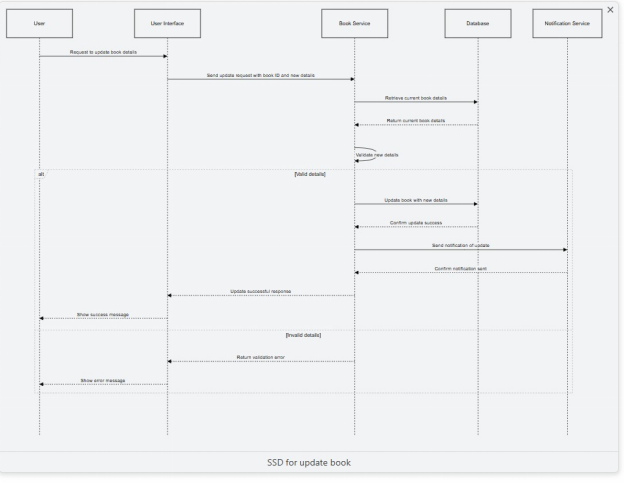
AI-generated content may be incorrect.

# Sequence Diagram:

A diagram of a book

AI-generated content may be incorrect.

# System Sequence Diagarm:



# Package diagram:

A diagram of a computer

AI-generated content may be incorrect.

# Fully dressed USE CASE: Add book:

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Use Case ID** | **UC002** |
| **Use Case Name** | **Add a Book** |
| **Brief Description** | **Allows an authorized user (Librarian or Bookstore Staff) to add a new book to the system by entering book metadata like title, ISBN, and author.** |
| **Primary Actor** | **Librarian / Bookstore Staff** |
| **Secondary Actor** | **Book Management System** |
| **Preconditions** | **User is logged in and authorized.- System is functional.- Data structure for book info exists.** |
| **Postconditions** | **Book saved in database.- Available for search, lending, or purchase.- Confirmation shown to user.** |
| **Trigger** | **User selects "Add Book" from the menu.** |
| **Basic Flow** | **1. User selects “Add Book”.2. System displays form.3. User fills details.4. System validates fields & ISBN.5. User submits.6. Data stored.7. Confirmation message.8. Book becomes searchable.** |
| **Alternative Flows** | **4a. Invalid Input: Show error; allow correction**  **4b. Duplicate ISBN: Show error; cancel or choose to update.** |
| **Exception Flows** | **System Failure: Show error; suggest retry.** |
| **Business Rules** | * **ISBN must be valid and unique.** * **Required fields must be filled.** * **Quantity must be non-negative.** |
| **Assumptions** | * **Data stored centrally.** * **Data entered is valid.** * **Cover image upload is supported.** |

# Class diagram:

A diagram of a book management system

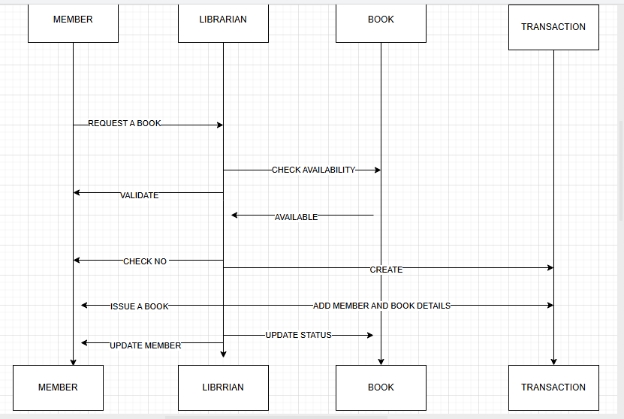
AI-generated content may be incorrect.

# Sequence diagram:

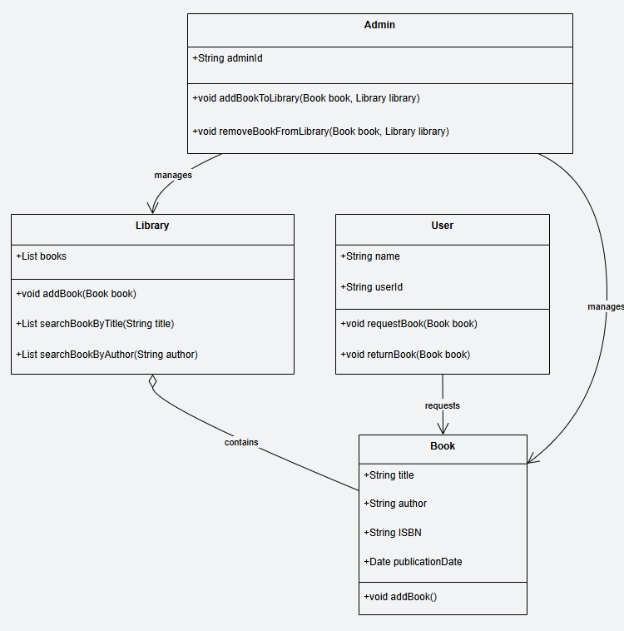
A diagram of a book

AI-generated content may be incorrect.

# System sequence diagram:



# Package diagram:



# System Event Design:

# 

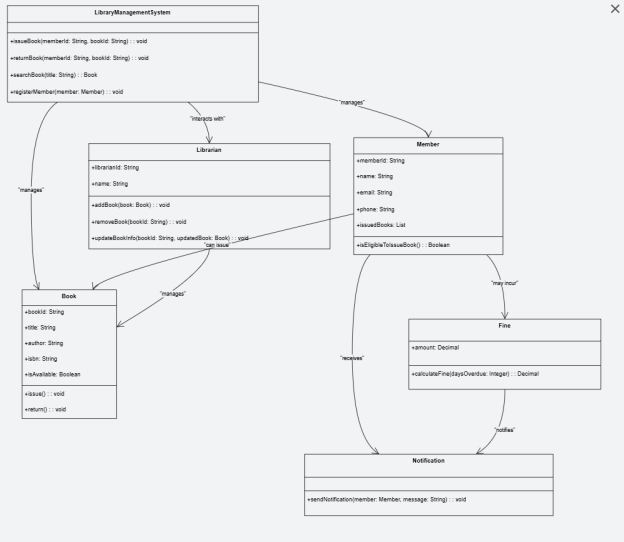
|  |  |  |
| --- | --- | --- |
| Actor Action (Trigger) | System Event | System Description / Response |
| User selects "Add Book" option | initiateAddBook () | System loads the "Add Book" form with relevant input fields |
| User fills in book details (Title, Author, ISBN, etc.) | — | Data entered manually — not a system event |
| User clicks "Submit" | validateBookInput(bookData) | System checks for required fields and validates ISBN format |
| (Optional) User uploads book cover image | uploadBookCover(imageFile) | System processes and stores image file with book record |
| System checks for duplicate ISBN | checkDuplicateISBN (ISBN) | System queries the database to see if the book already exists |
| System stores book information | storeBook(bookData) | Book is saved into the database if all validations pass |
| System displays success message | confirmBookAddition () | Confirmation is shown to the user that the book has been added |
| System makes book available for future operations | indexBookForSearch () | Adds the book to the catalog so users can search/lend/purchase it |

# ALISHBA RASHID SP23-BSE-022

# Fully Dressed Use Case: Issue Book

|  |  |
| --- | --- |
| Use Case Name | Issue Book |
| Use case ID | UC-01 |
| Primary Actor | Library Member(borrower) |
| Secondary Actors | Librarian |
| Brief Description | This use case describes the process by which a librarian issues a book to a registered library member. The system validates the member’s eligibility (no overdue books, no fines, and within borrowing limits) and checks the book's availability. If all conditions are met, the book is issued, and the due date is recorded. The librarian then hands over the book to the member. |
| Pre-conditions | Member is registered and active. - No outstanding fines or overdue books. - Book is available. - Librarian is logged into the system. |
| Post-conditions | · Book is marked as issued.  · Due date is recorded.  · Borrower’s account and book inventory are updated. |
| Trigger | A member requests to borrow a book. |
| Main Flow | · The librarian selects the “Issue Book” option in the system.  · The librarian enters or scans the member’s ID.  · The system validates the member by checking if they have overdue books or unpaid fines:  · The librarian enters or scans the book ID.  · The system checks if the requested book is available (not issued or reserved):  · The system records the book’s issue date and calculates the due date based on the member's type.  · The system marks the book as issued and updates the member’s account with the issued book.  · The system confirms the successful issuance of the book.  · The librarian informs the member of the due date and hands over the book to the member. |
| Alternative Flows | · 3a. Member has overdue books or unpaid fines   * 1. The system checks if the member has overdue books or unpaid fines.   2. The system detects that the member has overdue books or fines.   3. The system denies the book issue request.   4. The librarian informs the member of the issue (e.g., needs to clear fines before borrowing more books).   5a. Book is not available (already issued or reserved)   * 1. The system checks the availability of the book.   2. The system detects that the book is unavailable (e.g., already issued or reserved by another member).   3. The system informs the librarian that the book cannot be issued.   4. The librarian informs the member that the book is unavailable and may suggest placing a reservation for the book.   · 6a. Borrowing limit reached   * 1. The system checks if the member has reached their borrowing limit.   2. The system detects that the member has already borrowed the maximum number of books allowed.   3. The system denies the request to issue the book.   4. The librarian informs the member that they cannot borrow more books until they return some. |
| Business Rules | 1. The due date is calculated based on the member's type and library policy (e.g., students may have shorter loan periods than faculty). 2. The due date is automatically recorded when a book is issued. 3. Fines are applied for overdue books and must be paid before borrowing more books. 4. If a member has any unpaid fines, they are not allowed to borrow more books until the fines are cleared. 5. If a member has overdue books, they are not allowed to borrow additional books until the overdue books are returned. 6. The system performs real-time checks for overdue books, fines, and borrowing limits before issuing any book. |
| Exceptions | · If the librarian enters an invalid or unrecognized member ID, the system displays an error message and denies the request to issue the book.  · If the system encounters a failure (e.g., database connectivity issue), the librarian is informed, and the issue process is halted until the system is restored.  · If the requested book is already issued or reserved, the system informs the librarian that the book cannot be issued, and the member may be asked to place a reservation.  · If the member has overdue books or unpaid fines, the system denies the book issue request, and the librarian informs the member to resolve the issue before borrowing more books.  · If the member has reached their borrowing limit, the system denies the book issue request, and the librarian informs the member that they must return books before borrowing more.  · If the book's barcode is not scanned correctly or the system cannot find the book ID, the librarian is asked to rescan the book or manually enter the book's ID. |

# CLASS DIAGRAM:

****

System sequence diagram**:**

**A diagram of a book

AI-generated content may be incorrect.**

# Fullt dressed Use Case: Manage Members

|  |  |
| --- | --- |
| **Element** | **Details** |
| **Use Case Name** | **Manage Members** |
| **Use Case ID** | **UC-02** |
| **Primary Actor** | **Librarian** |
| **Secondary Actors** | **None** |
| **Brief Description** | **This use case describes how a librarian manages member records including registering new members, updating details, and deactivating or deleting members. The system validates inputs and ensures rules such as unique Member ID and borrowing restrictions are enforced.** |
| **Pre-conditions** | **- Librarian is logged into the system  - Librarian has permission to manage members** |
| **Trigger** | **Librarian selects the "Manage Members" option from the system menu** |
| **Post-conditions** | **- Member records are added, updated, or deactivated  - Changes are saved in the system and available for future transactions** |
| **Main Flow** | **1. Librarian selects "Manage Members"  2. System displays options: Add, Edit, Delete/Deactivate  3. Librarian selects desired action  4. System prompts for input  5. Librarian enters details  6. System validates inputs  7. System saves record  8. Confirmation shown** |
| **Alternative Flows** | **4a. Missing/Invalid Fields: System shows error, librarian corrects input  6a. Duplicate Entry: System detects duplicate ID/email, denies creation  5a. Active Loans/Fines: System blocks deletion, suggests deactivation** |
| **Business Rules** | **1. Each member must have a unique Member ID and email  2. Only active members may borrow books  3. Members with loans/fines cannot be deleted  4. Deactivated members are blocked from borrowing  5. All changes must be logged with timestamp and librarian ID** |
| **Exceptions** | **- Invalid Member ID entered: system shows error  - Database/system failure: system halts and informs librarian  - Member with active loans/fines: system denies deletion  - Improper input (e.g., format): prompts librarian to correct** |

# SYSTEM SEQUENCE DIAGRAMA white sheet of paper with black text AI-generated content may be incorrect.

# CLASS DIAGRAM: