GUL-E-NARJIS

SP23-BSE-023

USE CASE:

DELETE BOOK:

Use Case ID:	UC001
Use Case Name:	Delete Book
Brief Description:	This use case describes the process for deleting 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: i. Display an error message indicating the invalid credentials. ii. 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: i. Deny access to the "Update Book" functionality. ii. 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: i. Display an error message indicating that the book details could not be retrieved. ii. Prompt the user to try again later or provide an option to proceed with a partial view.

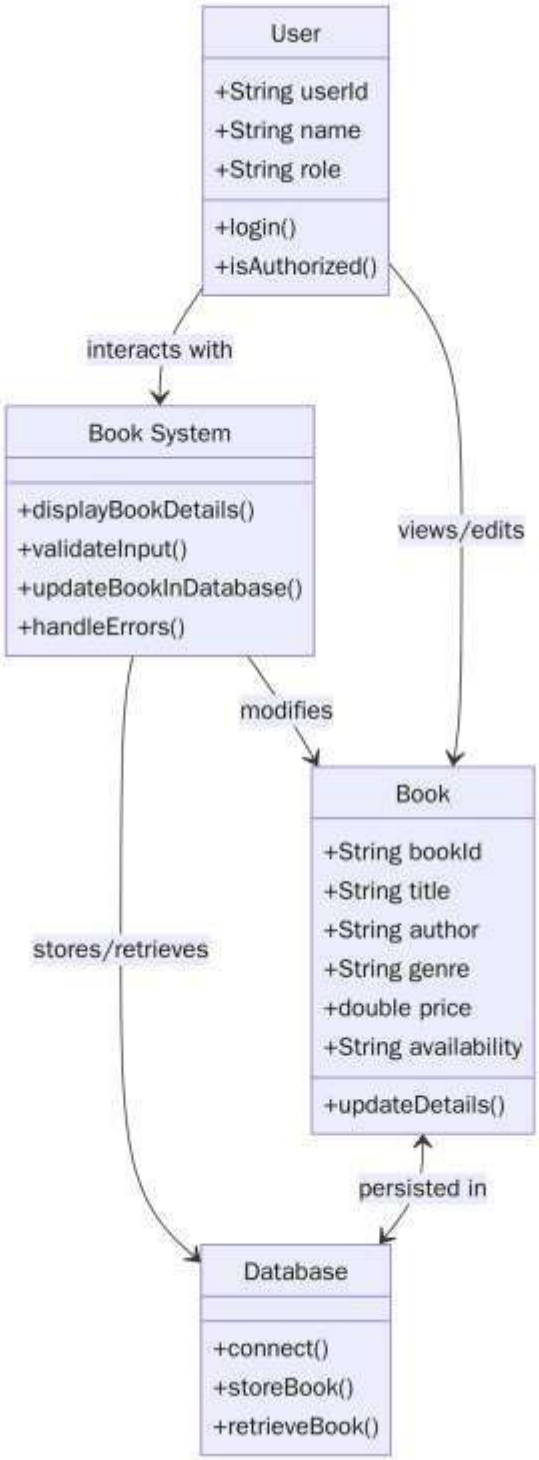
Exception Flows:	**a) Exception Flow 1 (Book Not Found)**: If the user tries to update a book that does not exist in the system, the system will: i. Display a message indicating that the book was not found. ii. Allow the user to select another book or exit the update process. **b) Exception Flow 2 (Network Failure)**: If the system loses connection to the database or encounters a network issue: i. Display an error message indicating the network issue. ii. Suggest the user retry the update once the connection is restored.
Business Rules:	i. A book's title must be unique in the system. Two books cannot have the same title. ii. The price of a book must be a positive number. iii. A book must be marked as either "available" or "unavailable."
Assumptions:	i. The system will handle concurrency issues (e.g., preventing two users from updating the same book simultaneously). ii. The system will maintain a backup of the original book information for auditing or recovery purposes.

System event design:

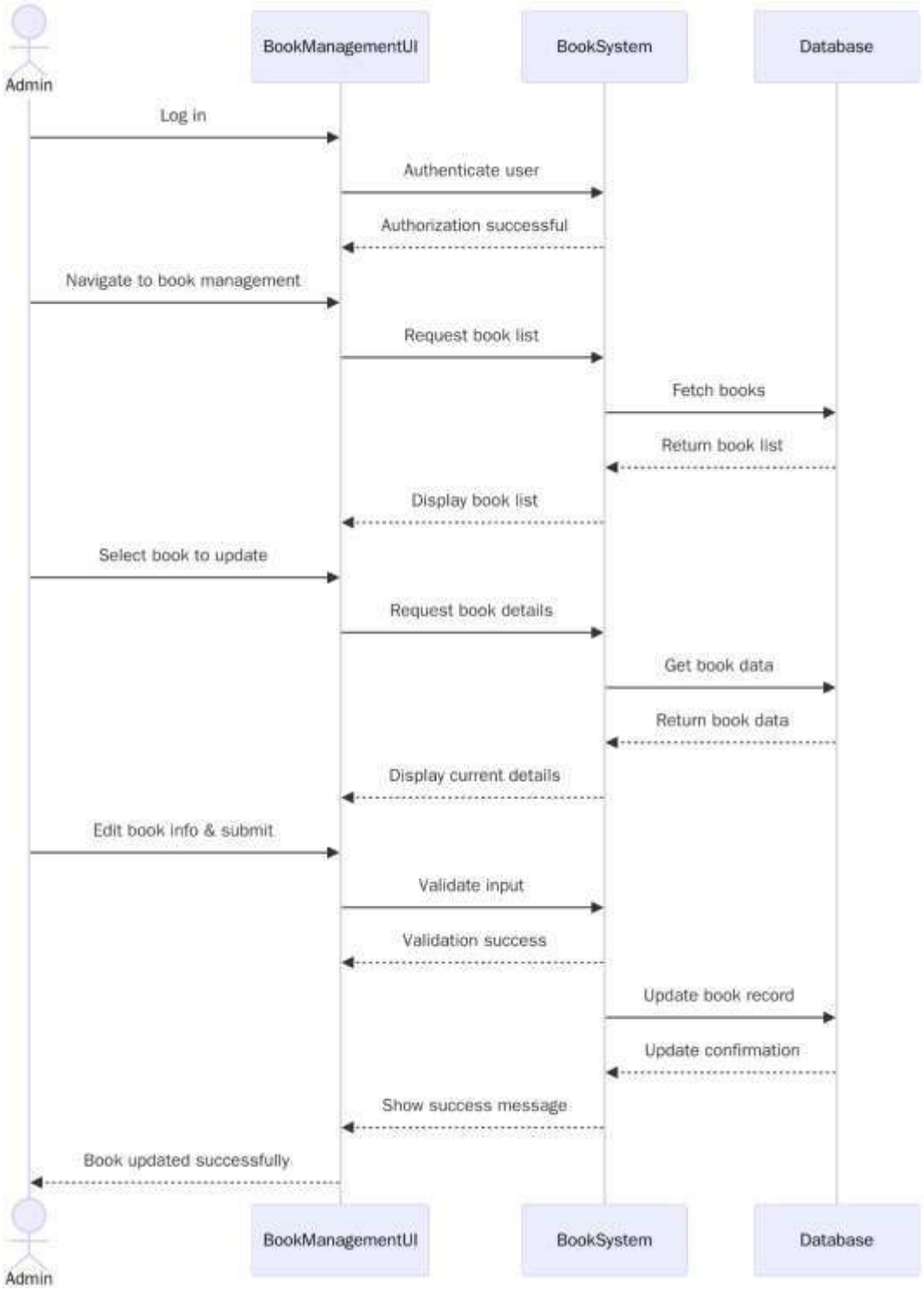
Actor Action (Trigger)	**System Event**	**Response/Outcome**
Selects "Update Book"	updateBookInfo(bookID)	System retrieves book details and displays them for editing.
Enters login credentials	authenticateUser (username, pwd)	System verifies credentials and sets access level.
Edits book details	submitUpdatedInfo(data)	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.

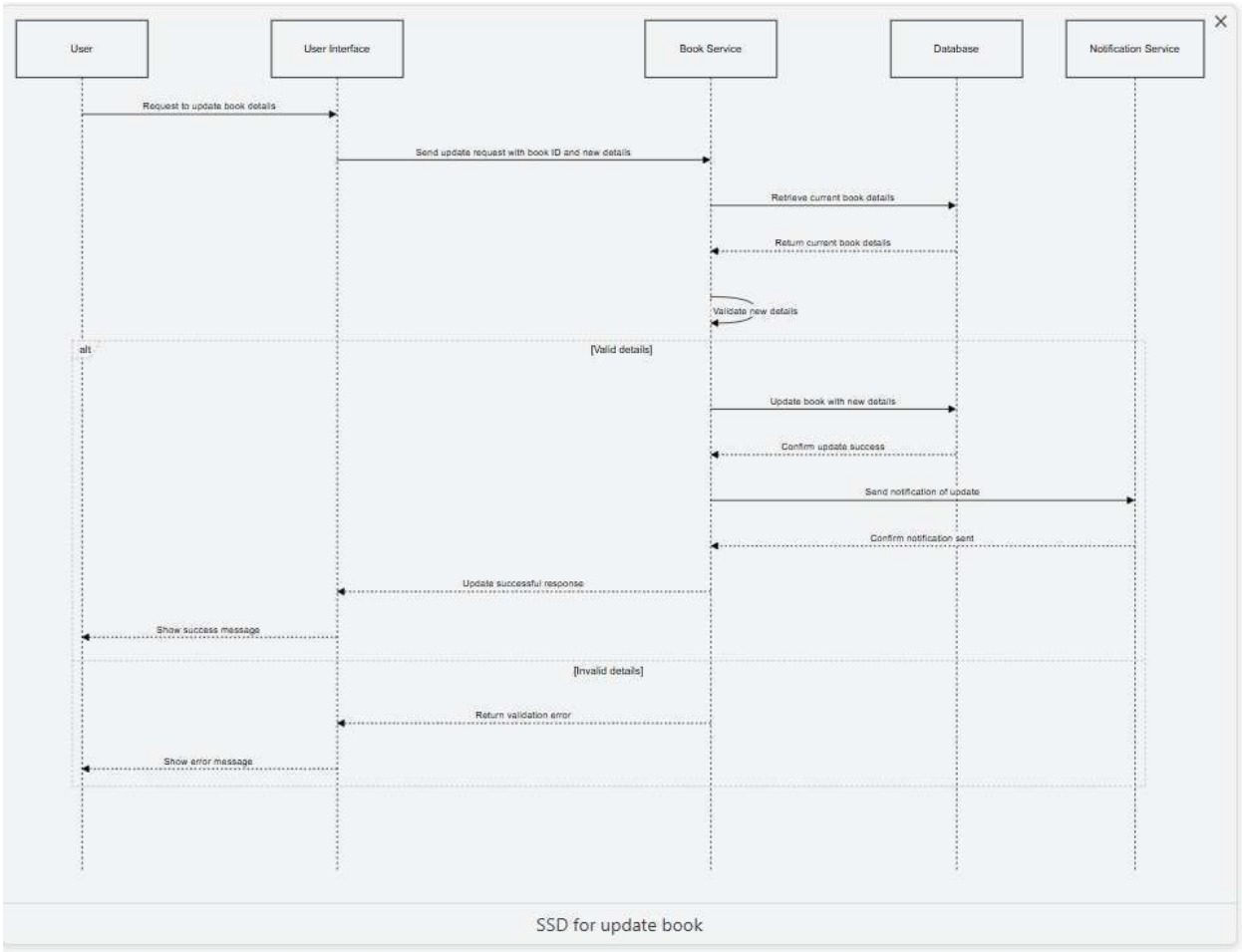
Class diagram :



Sequence Diagram:

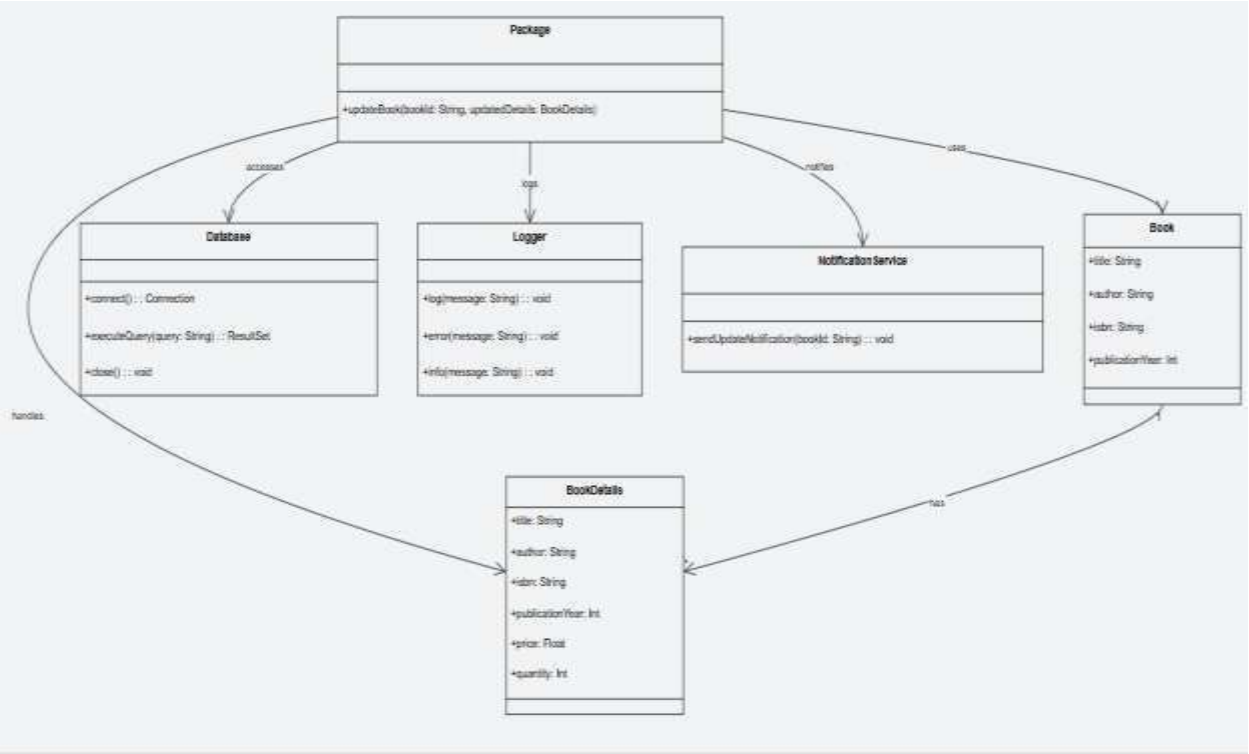

Admin
BookManagementUI
BookSystem
Database
Log in
Authenticate user
Authorization successful
Navigate to book management
Request book list
Fetch books
Return book list
Display book list
Select book to update
Request book details
Get book data
Return book data
Display current details
Edit book info & submit
Validate input
Validation success
Update book record
Update confirmation
Show success message
Book updated successfully
Admin
BookManagementUI
BookSystem
Database

System Sequence Diagarm:



SSD for update book

Package diagram:

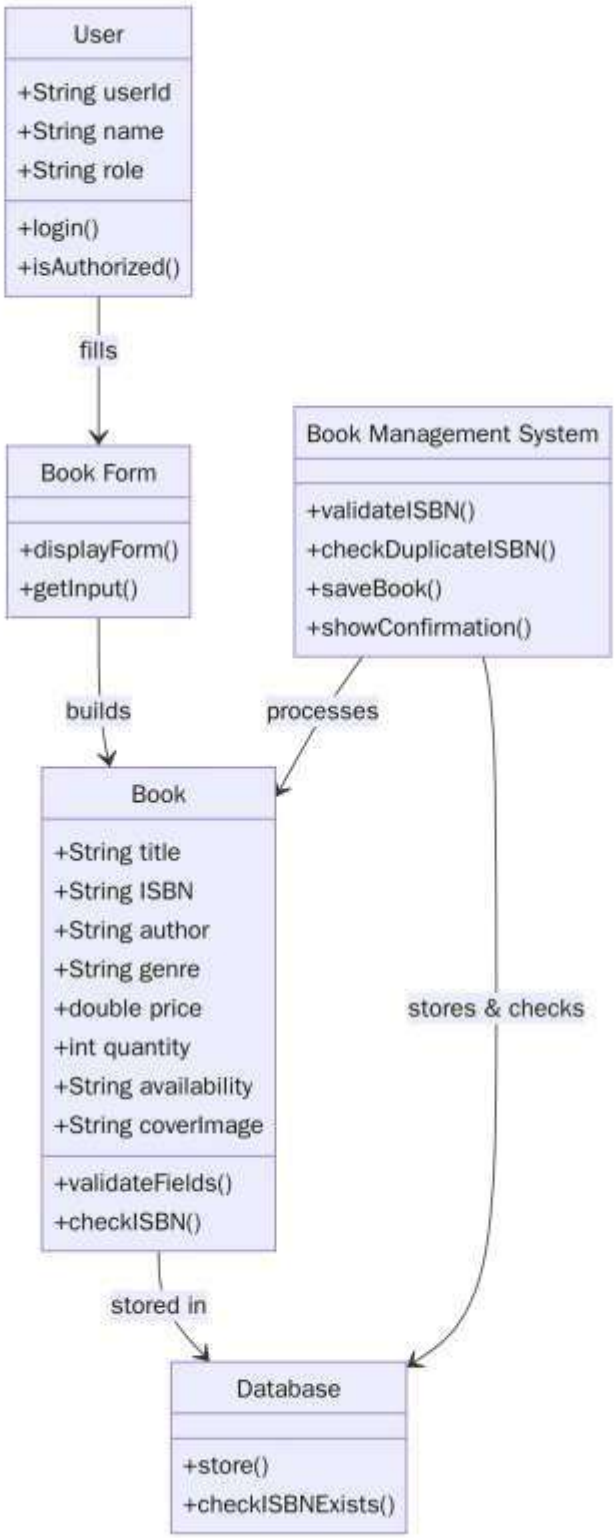

Package
+updateBook(bookId: String, updatedDetails: BookDetails)
accesses
logs
notifies
uses
Database
+connect() : : Connection
+executeQuery(query: String) : : ResultSet
+close() : : void
Logger
+log(message: String) : : void
+error(message: String) : : void
+info(message: String) : : void
Notification Service
+sendUpdateNotification(bookId: String) : : void
Book
+title: String
+author: String
+isbn: String
+publicationYear: Int
handles
BookDetails
+title: String
+author: String
+isbn: String
+publicationYear: Int
+price: Float
+quantity: Int
has

USE CASE:

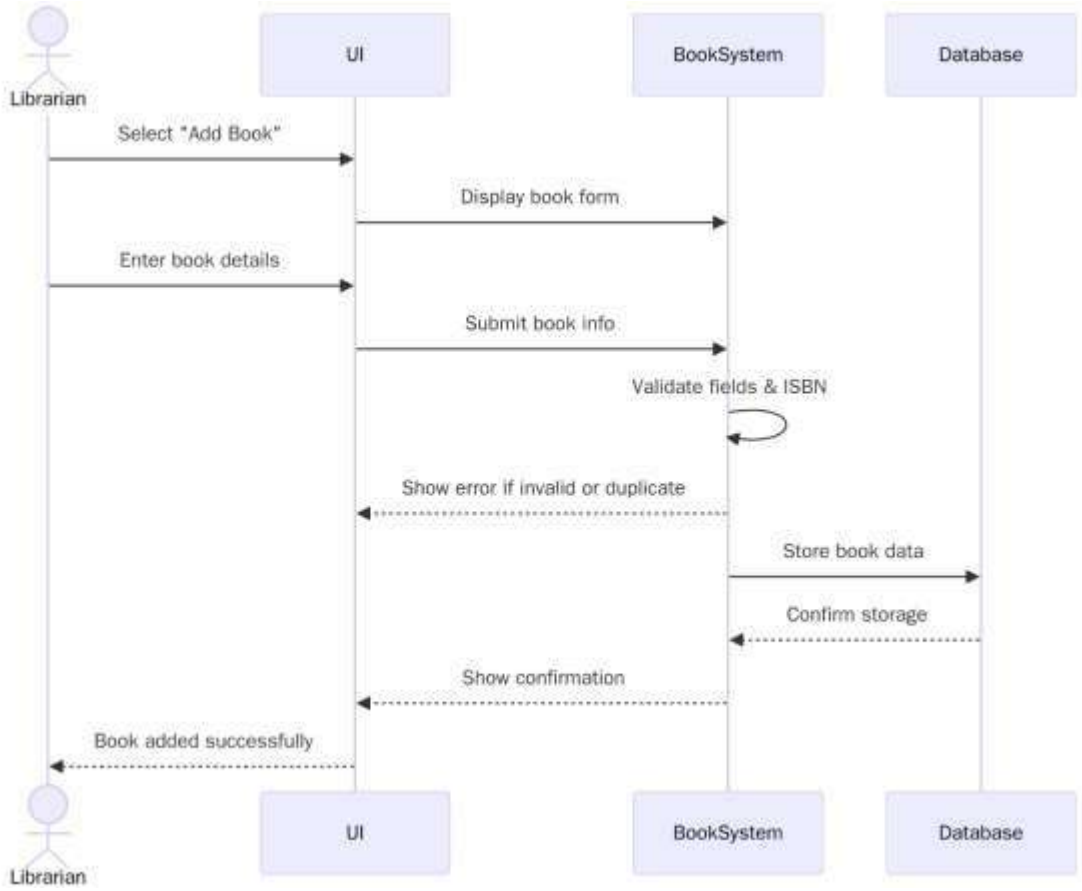
Add book:

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.**

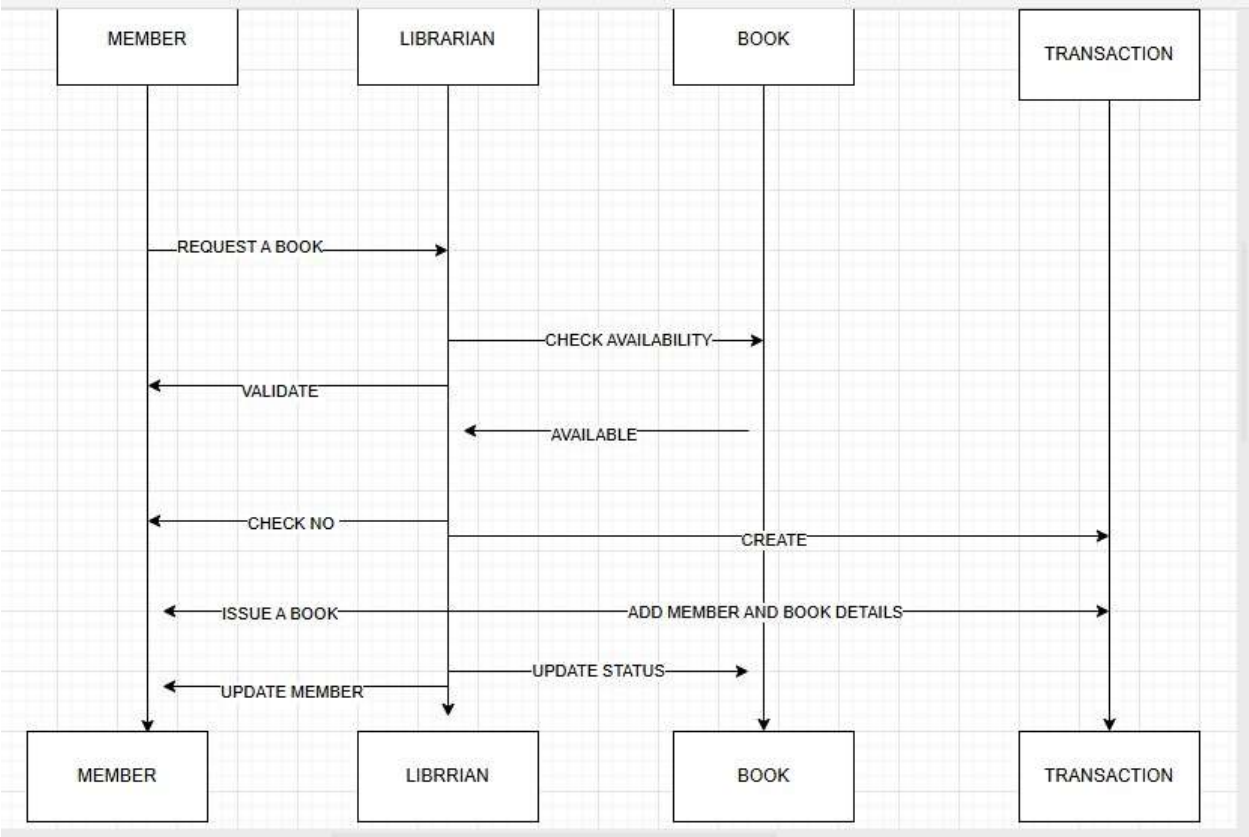
Class diagram:



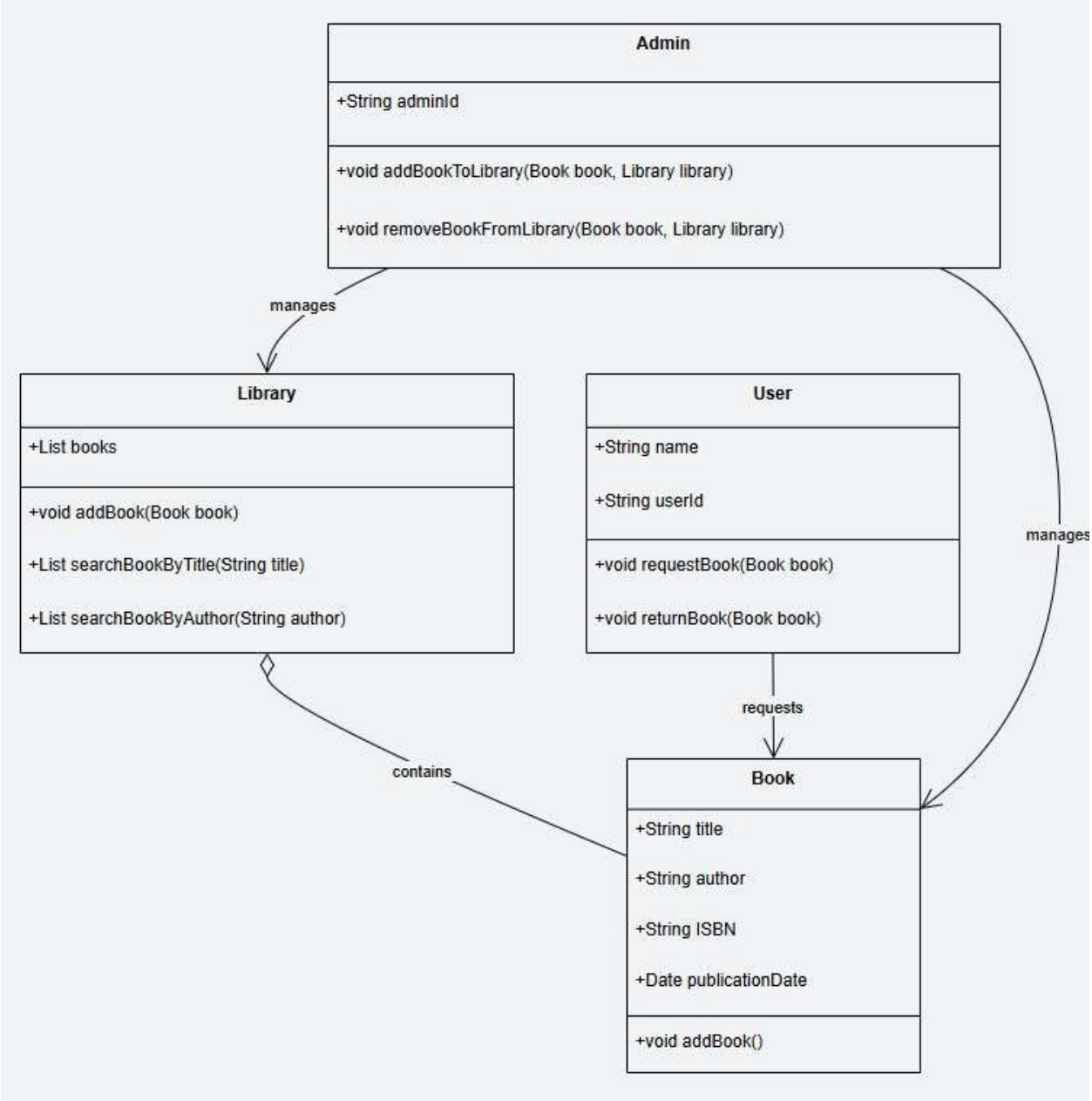
Sequence diagram:



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**