Library Management System & BiblioConnect

Integrated Library and Social Networking Platform

Introduction

- Library management system
  - Develop a library management system for librarians to use
    - Managing books, patrons and transactions etc…
- Create BiblioConnect, an integrated social platform that enhances user interaction with library resources.
  - Develop an integrated platform
    - Allows users to access library resources and promote social interaction.
    - As well as social activities that promote collaborative learning.

Project Goals

- **Library Management System Goals:**
  - Efficient Book Management: Adding, updating, and removing books; searching and listing available books.
  - Effective Patron Management: Registering and managing library members.
  - Transactions: Borrowing and returning books with tracking and reporting.
- **BiblioConnect Goals:**
  - Community Building: Facilitate social interactions and create a vibrant community.
  - User Engagement: Encourage discussions, sharing, and real-time interactions.
  - Convenience: Combine library services with social networking features.

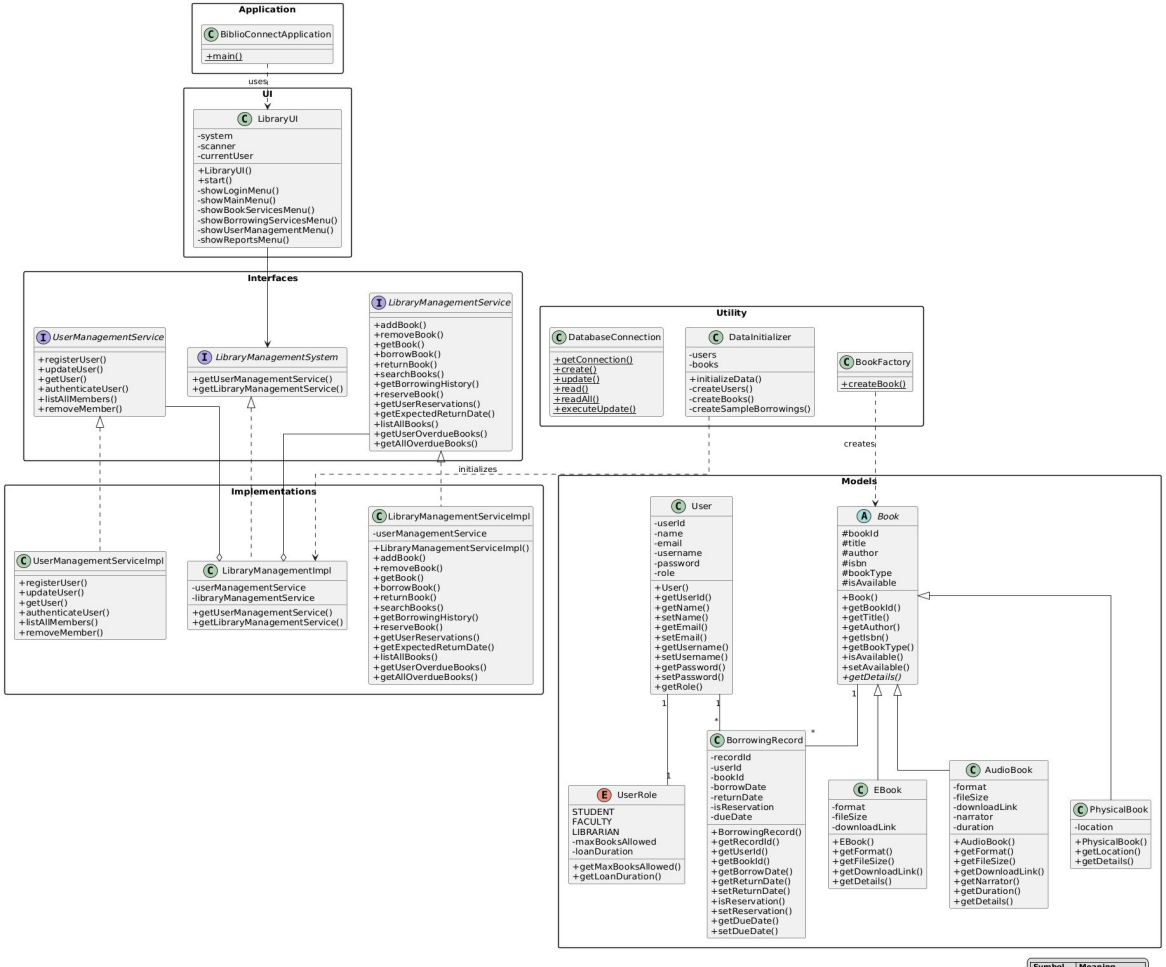
Library Management System Overview

- **Core Functionalities:**
  - **Book Management:** Create, Read, Update, Delete on books.
  - **User Management:** Functionality for library members.
  - **Transaction Management:** Borrowing and returning books, tracking due dates.
  - **Reporting:** Generate details on borrowed and overdue books.
  - **Design Approach:** Use of OOP principles to ensure modular and maintainable code.

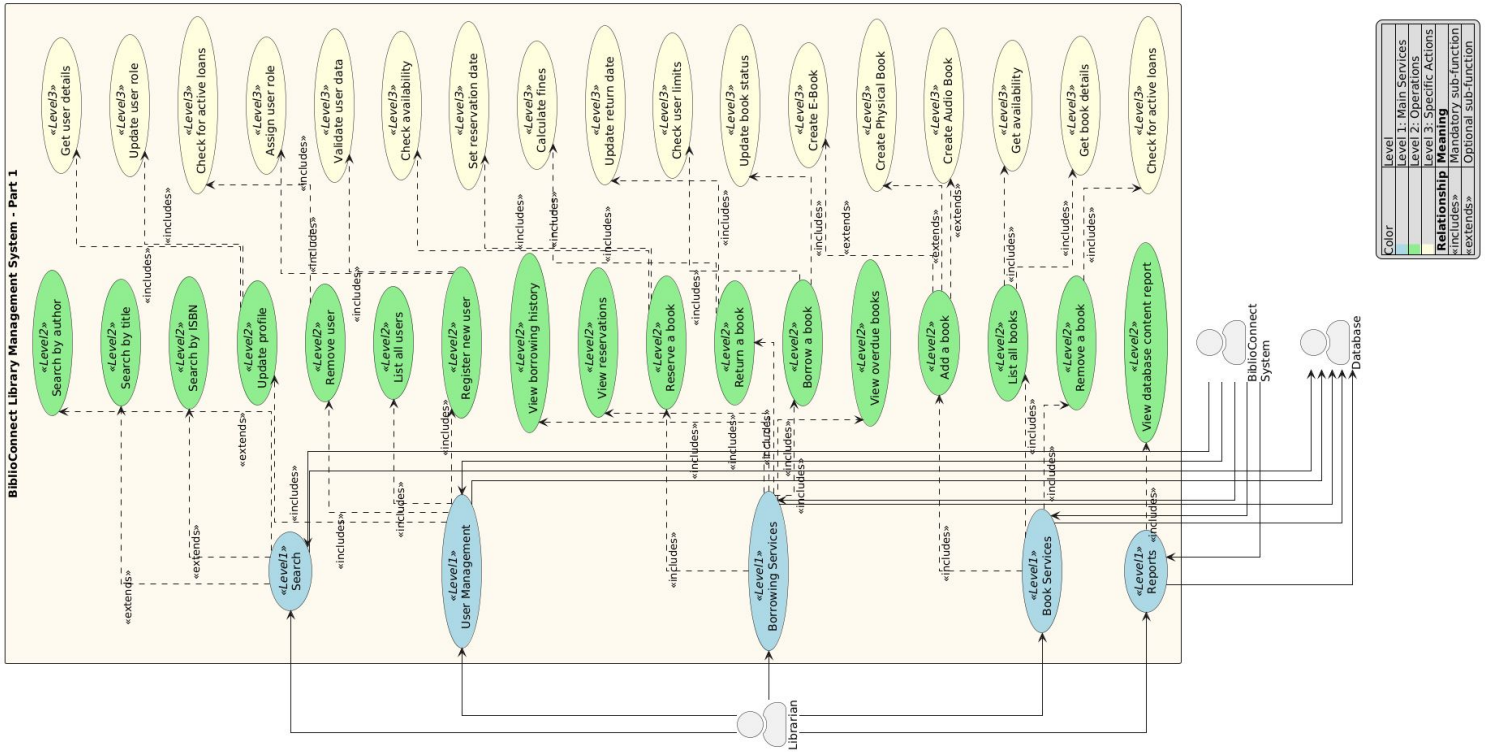
UML Design Overview

- **High-Level UML Diagrams:**
  - **Class Diagram:** Showing main classes like LibraryUI, UserRoles, LibraryManagement, and their relationships.
  - **Using Case Diagram:** Showcases main interactions like adding a book, borrowing a book, etc.
  - **Sequence Diagram:** Describes the sequence of actions for key operations stated and more.

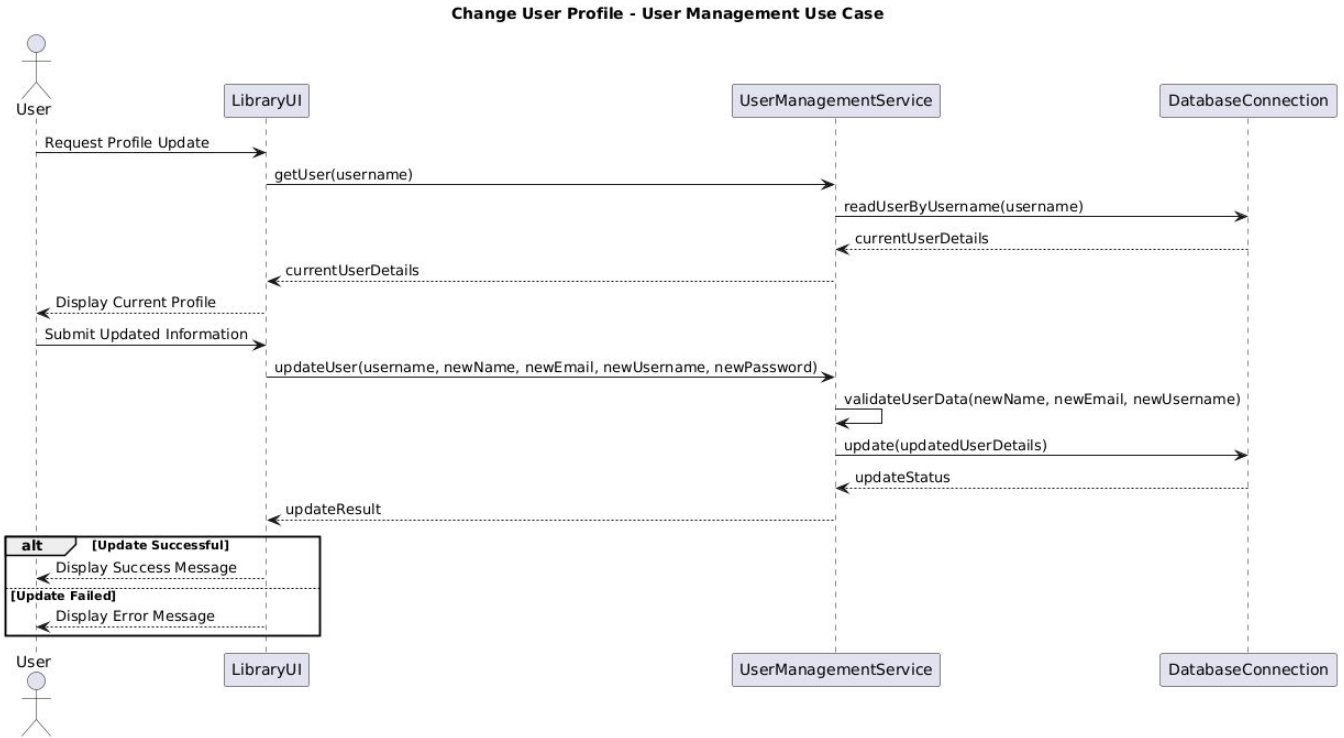
Class Diagran



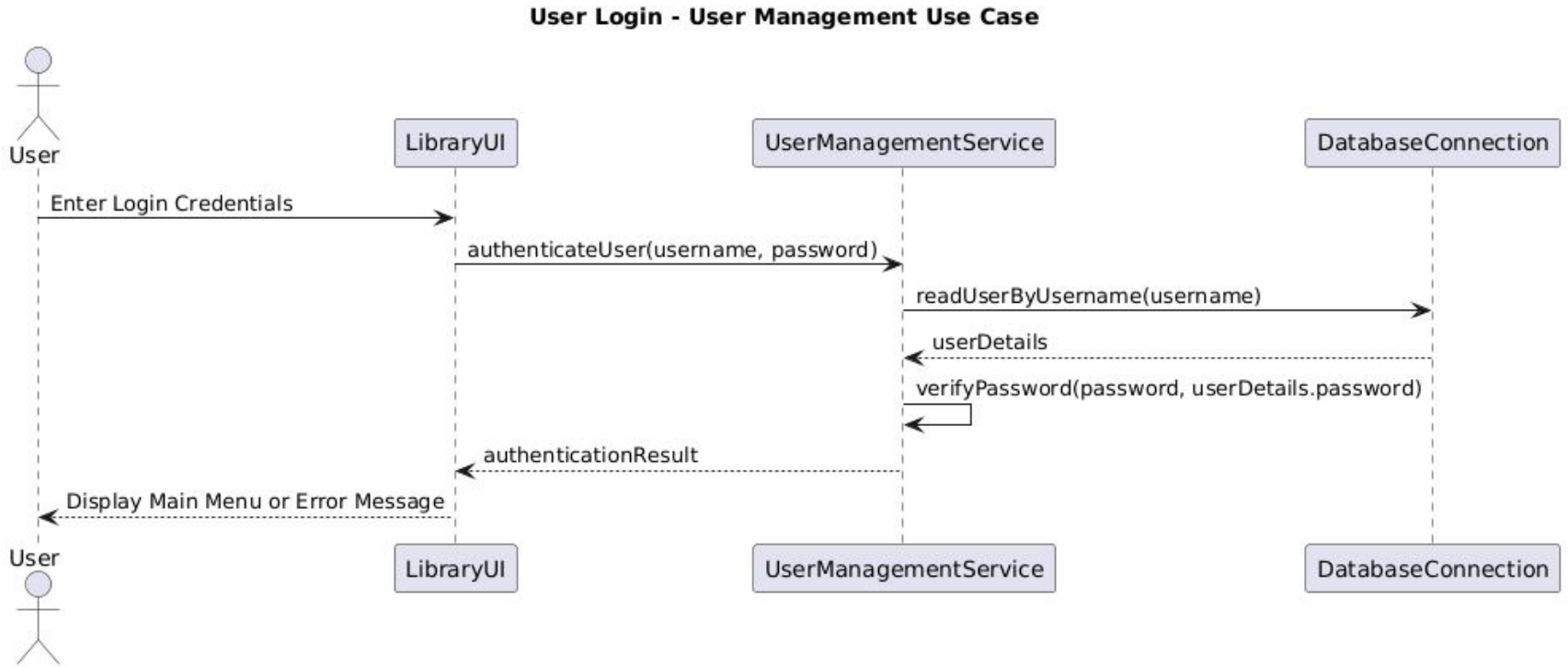
Use Case diagram



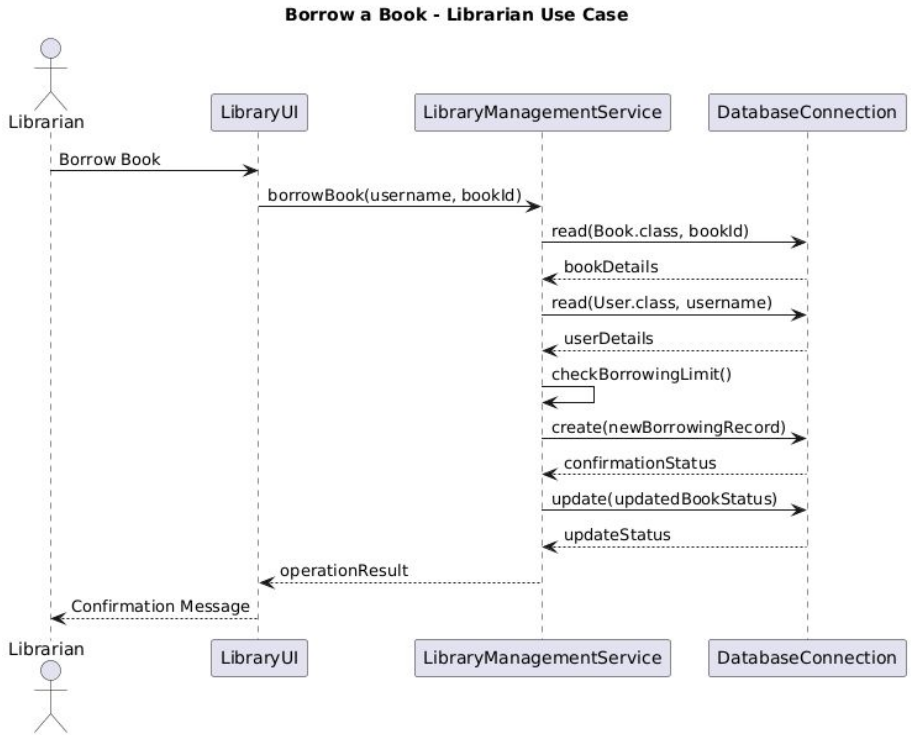
Sequence Diagrams

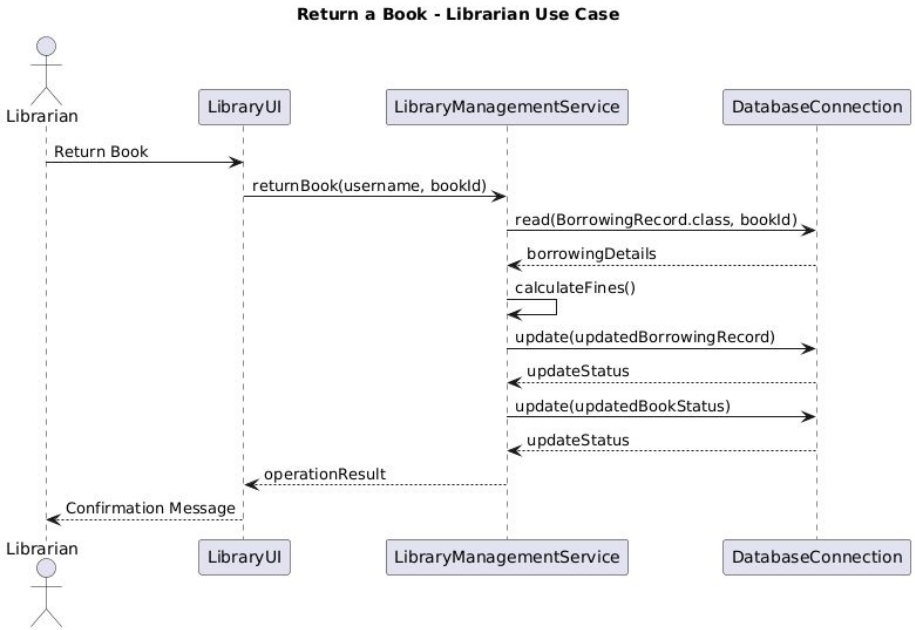


Sequence Diagrams

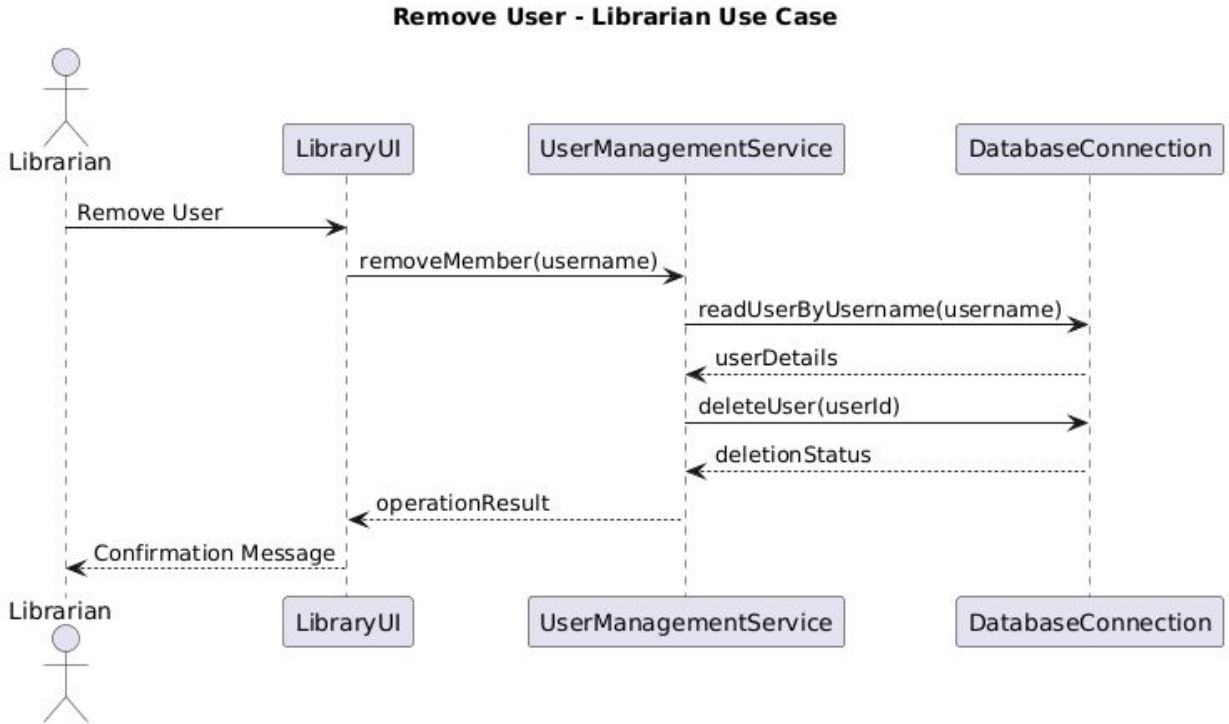


Sequence Diagrams





Sequence Diagrams



Book Management Features

- **Adding Books:**
  - Input details: Title, Author, ISBN, etc.
  - Stored in the system catalog.
- **Removing Books:**
  - Search and delete book records.
- **Updating Books:**
  - Modify details of existing book records.
- **Searching Books:**
  - Search by title, author, or ISBN.

Patron Management Features

- **Registering New Members:**
  - Inputting necessary details to create a new member
- **Removing Members:**
  - Deleting records of a member when no longer needed.
- **Updating Member Information:**
  - Edit details such as email, password, user role, etc...
- **Listing Registered Members:**
  - Display all members with their details.

Borrowing and Returning Books Feature

- **Borrowing Process:**
  - Check out books to members, update the status to borrowed
- **Returning Process:**
  - Return books, update status to available, and check for any overdue fines.
- **Preventing Overdue Borrowing:**
  - Verifying books that are checked out and ensuring they are not able to be borrowed again until returned.

Reporting Features

- **Report Generation:**
  - Create a report on number of users, books, borrowed books and overdue books
- **Benefits:**
  - Help manage inventory and users

User Interface

- **User experience:**
  - Simple and effective navigation
  - Clear prompts that provide helpful feedback for users
- **Features:**
  - Main Menu: Options for adding books, borrowing books, viewing reports, etc.
  - Book Management: Interfaces for adding, removing, and updating books.
  - Patron Management: Interfaces for managing member information.
  - Transaction Management: Interfaces for borrowing and returning books.

OOP concepts incorporated

- Inheritance:
  - Example: Create unique member types like Student and Teacher inheriting from a base member class.
  - Purpose: Reuse common attributes and methods while adding specific features for each type.
- Composition:
  - Example: Manage the relationship between Book and Author classes using composition.
  - Purpose: Ensure that Book instances have associated Author instances, reflecting real-world relationships.
- Abstraction:
  - Example: Define abstract classes or interfaces like LibraryManagement for common operations (e.g., borrowing, returning).
  - Purpose: Hide implementation details and expose only necessary functionalities.
- Interfaces:
  - Example: Use interfaces such as Borrowable to define common behaviors for borrowable items.
  - Purpose: Allow flexibility and scalability for adding new types of borrowable items and other future items.

Testing and cases

- **Testing Approach:**
  - Unit Testing: Test individual components like Book, members, and Transaction classes
  - Integration Test: Observing interactions between classes, like borrowing and returning books.
  - User Interface Testing: Ensure UI responds as expected to user inputs and displays correct feedback information.
- **Key Tests Performed:**
  - Adding and removing books/patrons.
  - Borrowing and returning books with due date management.
  - Generating accurate reports on borrowed and overdue books.
  - Corner cases and edge cases tested thoroughly

Part 2 - BiblioConnect

- Integrated Platform:
  - Objective**:** Build on the Library Management System to create a social networking platform for readers.
  - Features**:** Combine library functionalities with social interactions to enhance user experience.
- Core Features:
  - User profiles, social interactions, groups and discussions, following mechanisms, and events.
- Design Considerations:
  - Ensure seamless integration with the existing library management system.
  - Create a user-friendly and engaging interface.

User Profiles

- Profile features
  - **Favorite Books:** Users can showcase their favorite books on their profiles.
  - **Reading Habits:** Display reading statistics and preferences.
  - **Literary Preferences:** Allow users to indicate genres or authors they are interested in
- Benefits
  - Personalize user experience and create connections based on reading interests.

Social Interactions and Features

- Interaction Features:
  - Posting Messages: Users can create updates, thoughts, and reviews about books.
  - Commenting: Allow users to comment on posts and engage in discussions.
  - Liking and Sharing: Users can like or share content, allowing for engagement with other users
- Benefits:
  - Community engagement and interactions with readers.

Groups and Discussions

- Groups:
  - Groups based on interest: Users can join or create groups based on specific interests.
  - Group management: Features for group admins to manage membership and content.
- Discussions:
  - Topic-Based Discussions: Participate in conversations about various topics.
  - Community Engagement: Encourage active participation and sharing.
- Benefits:
  - Social interaction and community bonding, promotes collaborative learning.

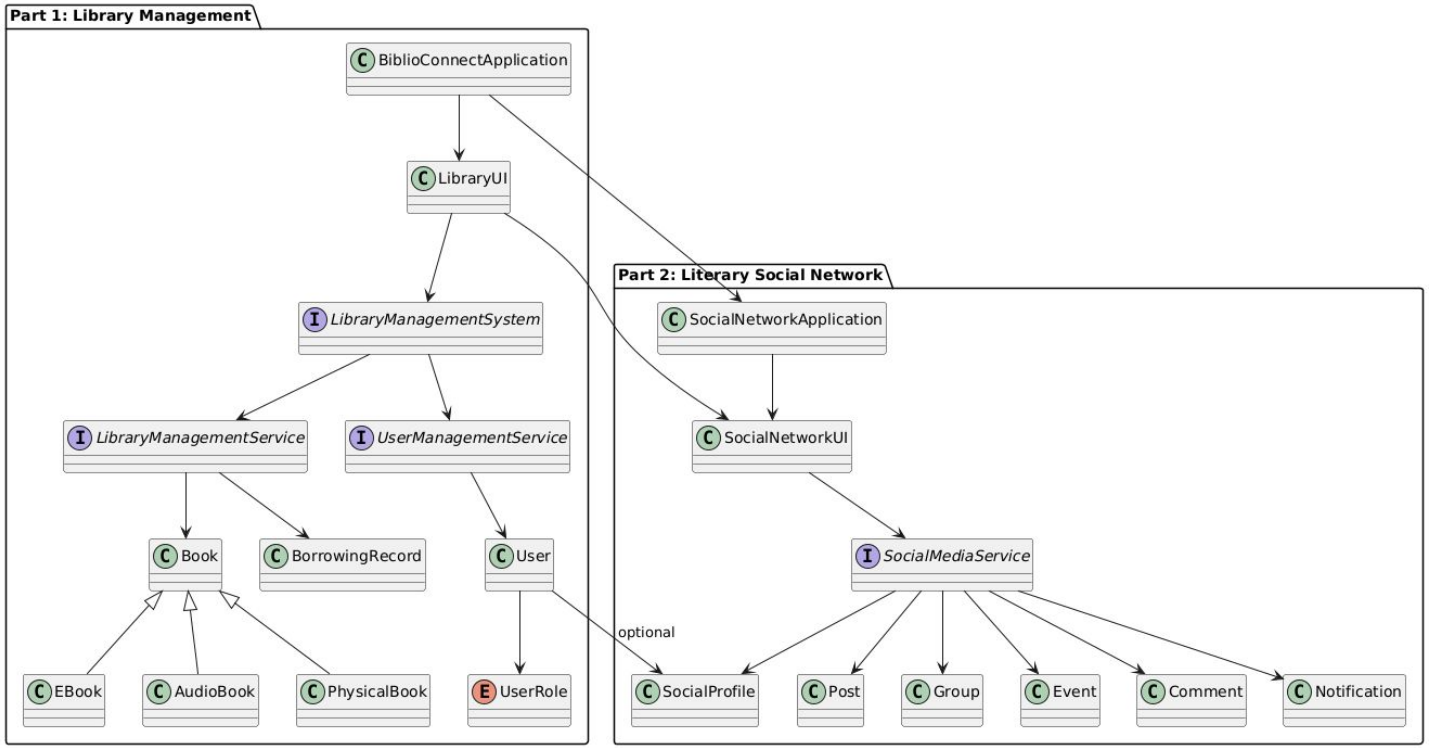
Following Feature

- Following Users: Users can follow others to see updates and activities.
  - Receiving Updates: Stay informed about followed users' book recommendations and posts.
  - Builds connections that promote networking and relationship building with fellow users.
- Benefits:
  - Promotes a dynamic and collaborative experience for users.

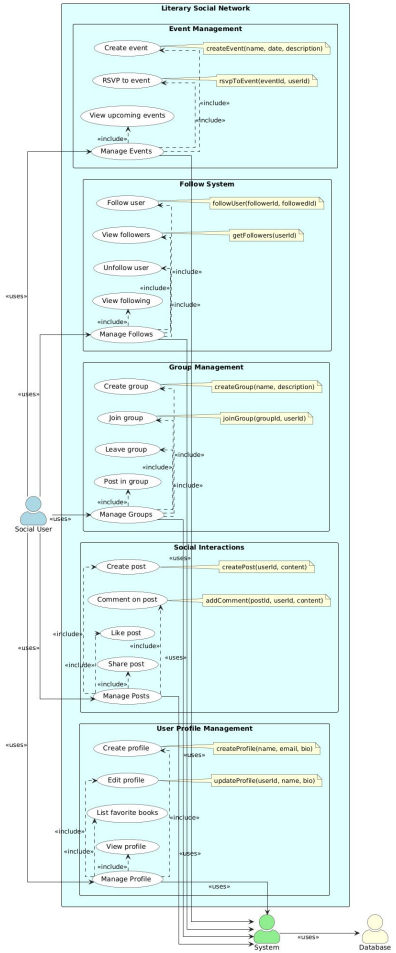
Events and Meetups

- Event Features:
  - Literary Events: Host book clubs, author signings, and literary festivals.
  - Event Management: Users can create, join, or RSVP to events.
- Meetups:
  - Organizing Meetups: Create in-person or virtual meetups for book discussions or author interactions.
  - Community Participation: Encourage users to participate in community events.
- Benefits:
  - Community bonding and social networking with like minded individuals.

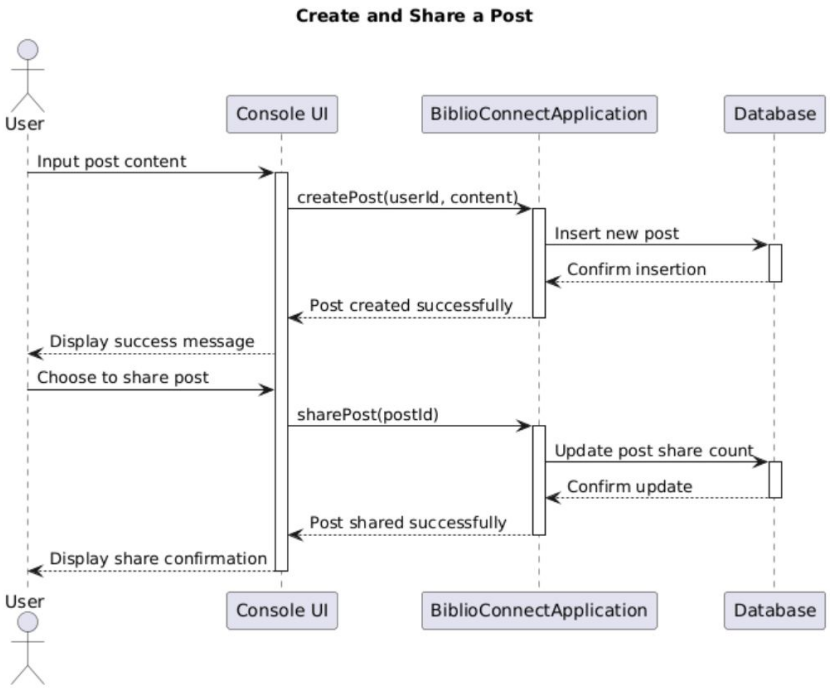
Class Diagram



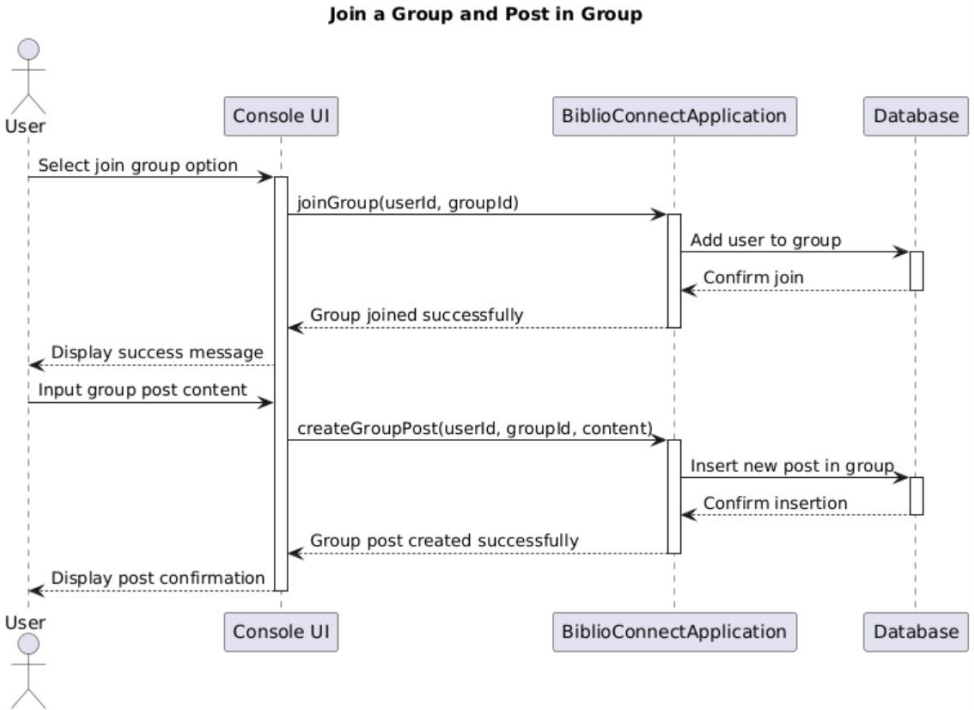
Use Case Diagram



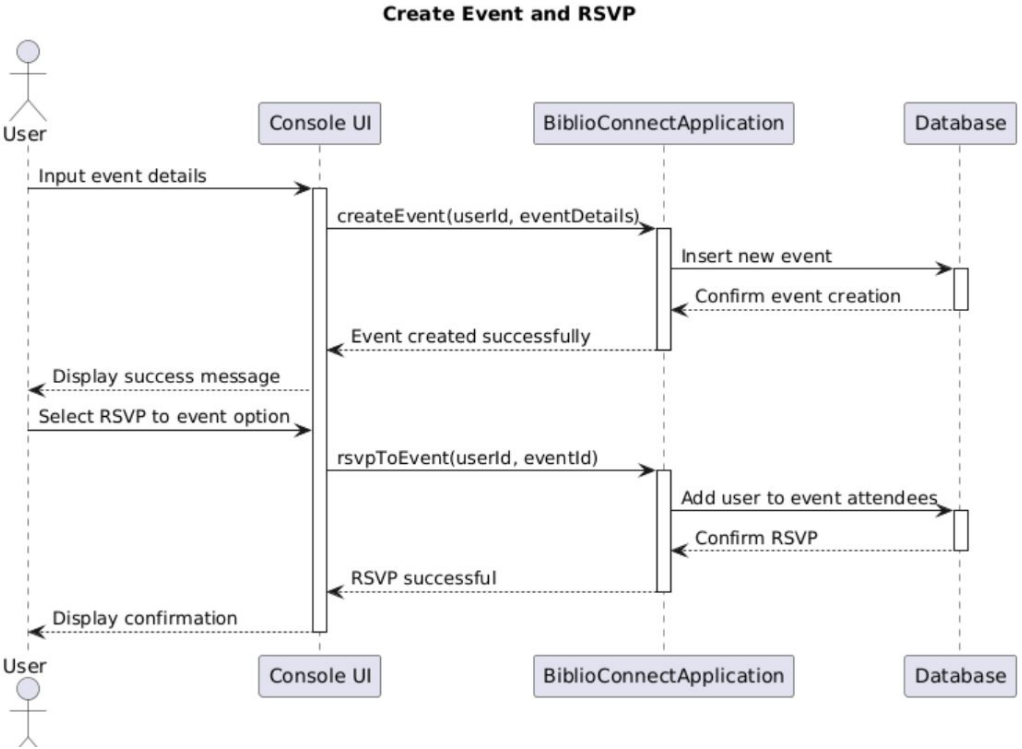
Sequence Diagrams



Sequence Diagrams



Sequence Diagrams



Integration

Modular Architecture: The system is designed with two distinct but interconnected modules - Library Management and Literary Social Network.
Minimal Changes to LIbrary Management Sytem: The existing Library Management System remains largely unchanged, with integration points added at the application and UI levels.
Separate Databases: The two maintain separate databases to ensure independence and scalability.
User Integration: Library users (from Part 1) can optionally create social profiles in Part 2, enabling single sign-on across both systems.
Application-Level Integration: BiblioConnectApplication serves as the central point, connecting both the Library and Social Network components.
UI Integration: LibraryUI links to SocialNetworkUI, allowing users to access social features from the library interface. Book Reference System: Literary Social Network implements a BookReference class to discuss books without full synchronization with the library database.
On-Demand Data Fetching: Basic book information is fetched from only when needed, using a limited API.
Extensible Design: The use of interfaces and modular architecture allows for easy future enhancements in both parts.
User-Driven Content: The social network allows discussions about any book, whether in the library or not, promoting a flexible and engaging literary community.

Key Learnings

The project was refactored at least 8-10 times to ensure OOP concepts learnt in class.

Doing a design that follows OOP helped later when changes were to be made or extensions were planned to introduce a social media platform

Design and Testing/Fixing bugs took more time than coding

A proper class design is essential to a good software project. However it cannot be achieved upfront and requires iteration.

It requires a lot of diligence and effort to stick to OOP principles, especially when bug fixing because the impulse is to fix bugs by patching code

Defensive design to ensure corner cases are address have to be part of requirements, else the code will not work properly.

Among UML diagrams, Class diagrams are very important as they help visualize if OOP principles are being followed. Use case diagrams are useful for customer communication. Sequence diagrams are more implementation specific.

OOP Principles in this project

This project implements several key software design principles and best practices:

1. SOLID Principles: a. Single Responsibility Principle (SRP):
    - Each class has a single, well-defined responsibility (e.g., User class manages user data, LibraryUI handles user interaction).
    - Services are separated into distinct interfaces (UserManagementService, LibraryManagementService).

2. b. Open/Closed Principle (OCP):
    - The Book class is abstract and can be extended without modifying existing code (PhysicalBook, EBook, AudioBook).
    - New user roles can be added to the UserRole enum without changing user management logic.
    -
3. c. Interface Segregation Principle (ISP):
    - Separate interfaces for user management and library management allow clients to depend only on the methods they use.
    -
4. d. Dependency Inversion Principle (DIP):
    - High-level modules (e.g., LibraryUI) depend on abstractions (interfaces) rather than concrete implementations.
    -
5. Cohesion and Coupling: a. High Cohesion:
    - Classes have closely related responsibilities (e.g., UserManagementServiceImpl focuses solely on user management operations).
    - Methods within classes are strongly related to the class's purpose.
    -
6. b. Low Coupling:
    - Use of interfaces (e.g., LibraryManagementSystem) reduces dependencies between components.
    - Dependency injection is used to provide services to classes that need them.
    -
7. Extensibility: a. The factory pattern (BookFactory) allows easy addition of new book types. b. The system architecture allows for easy addition of new features or services. c. Use of enums (UserRole) makes it simple to add new user roles with specific privileges.

Conclusion

- Successful goals:
  - Development of a comprehensive library management system
  - Robust social platform with social features
- Future possibilities
  - Additional features: More advanced features that allow for more functionality for users.
  - Improve scalability that would allow for more users and a more complex library.