Software Architecture Requirement and Design

Cloud Library

Team - 6

Ganne Tejaswi – 8

Swetha Chandra Karroti – 17

Pallavi Ramineni – 31

Sampoorna Swathi Gorantla – 11

Design an application that is a Cloud Library. In essence the application must allow the user to upload the books and then do basic operations on the document like editing the pages, sharing the books to other users. The application must allow users to save their favourite books list and also suggest books to other readers. The functionality of the operations must be done in the form of APIs. Try to use the cloud infrastructure in this application.

The basic functionality of this application is that it must allow users to upload the books and perform basic operations on the document. The operations include editing the pages, sharing the books to other users, save their favourite books list and also suggest books to other readers.

So, here is the Use case diagram, Class diagram, Activity and sequential diagrams.

Library application uses RESTful APIs. For example, if any book is unavailable with the library management, they will use the Google Books API to search and lookup for an e-book. The application communicates with the database to search for the available books, add a new book or delete the books. The user can also perform the operations like sharing the book, suggesting a book or save their favourite books.

Functionality of each component:

Library Application- This is a user friendly application. Here the user can do the following operations

- Suggest a book to other user
- Share a book
- Save their favorite book list
- Edit the document

Database- Storage and the retrieval of books can be done in and from the database. All the user information can also be stored.

RESTful API- It is connected to the database. When the user requests for certain data/book, it can be accessed from the database using RESTful API.

Layered Architecture:

1.Presentation tier

2.Application tier

3.Database tier

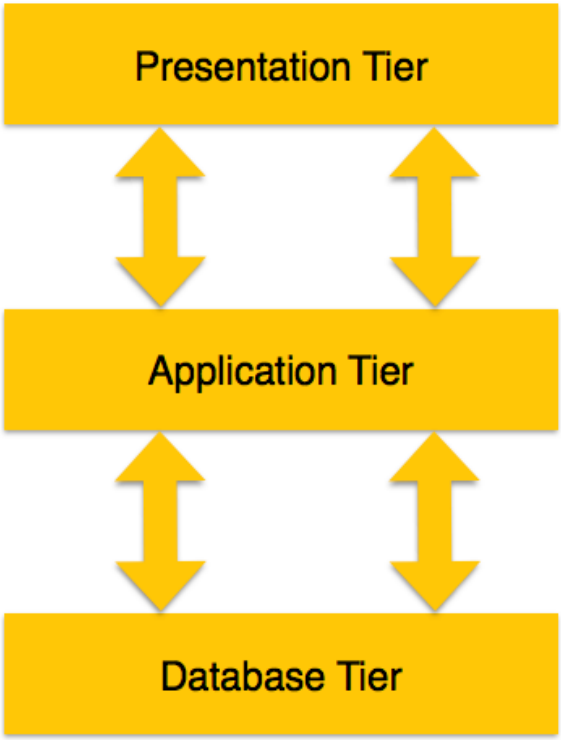
An Architectural style suitable for cloud library use case is the **Layered Architectural style**.

Layered architectural style divides the functionality of the application into separate layers. Explicit and loosely coupled communication exists between the layers in the layered architecture.

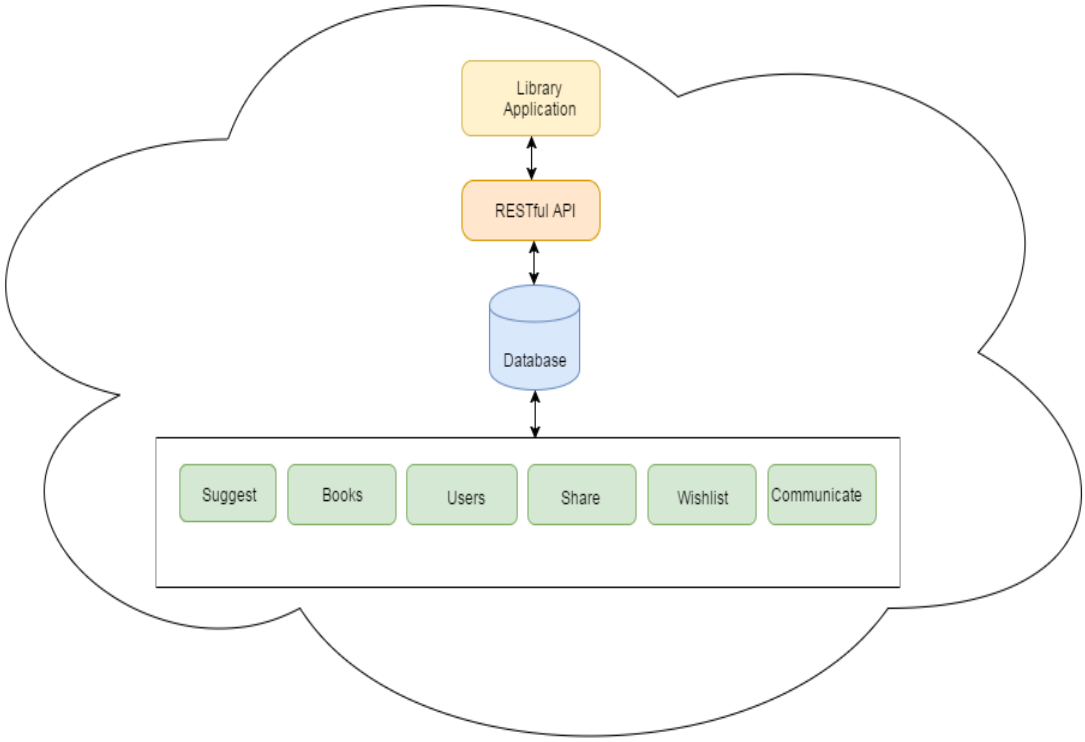
In the case of cloud library use case, we had divided the architecture into presentation layer which depicts the view of the system, Application layer which provides the required REST services and Data Access layer which supports networking, data storage e.t.c. In library, operations like editing the pages, deleting or updating of the pages takes place in the data access layer which can be shared with the application layer by the help of services provided by the service layer. Users can interact with the application by the application layer and can access the books in the library data store by using API services. They can also save their favorite books list in the data store or share their favorite books with other users by Library API services provided by the application layer. Uploading the books can be done by one user and other users may access those books from the data store. As these operations are all using services layer to interact with the data access layer, we thought that layered architecture is the perfect architectural style for the cloud library.

Layered architectural style had several advantages

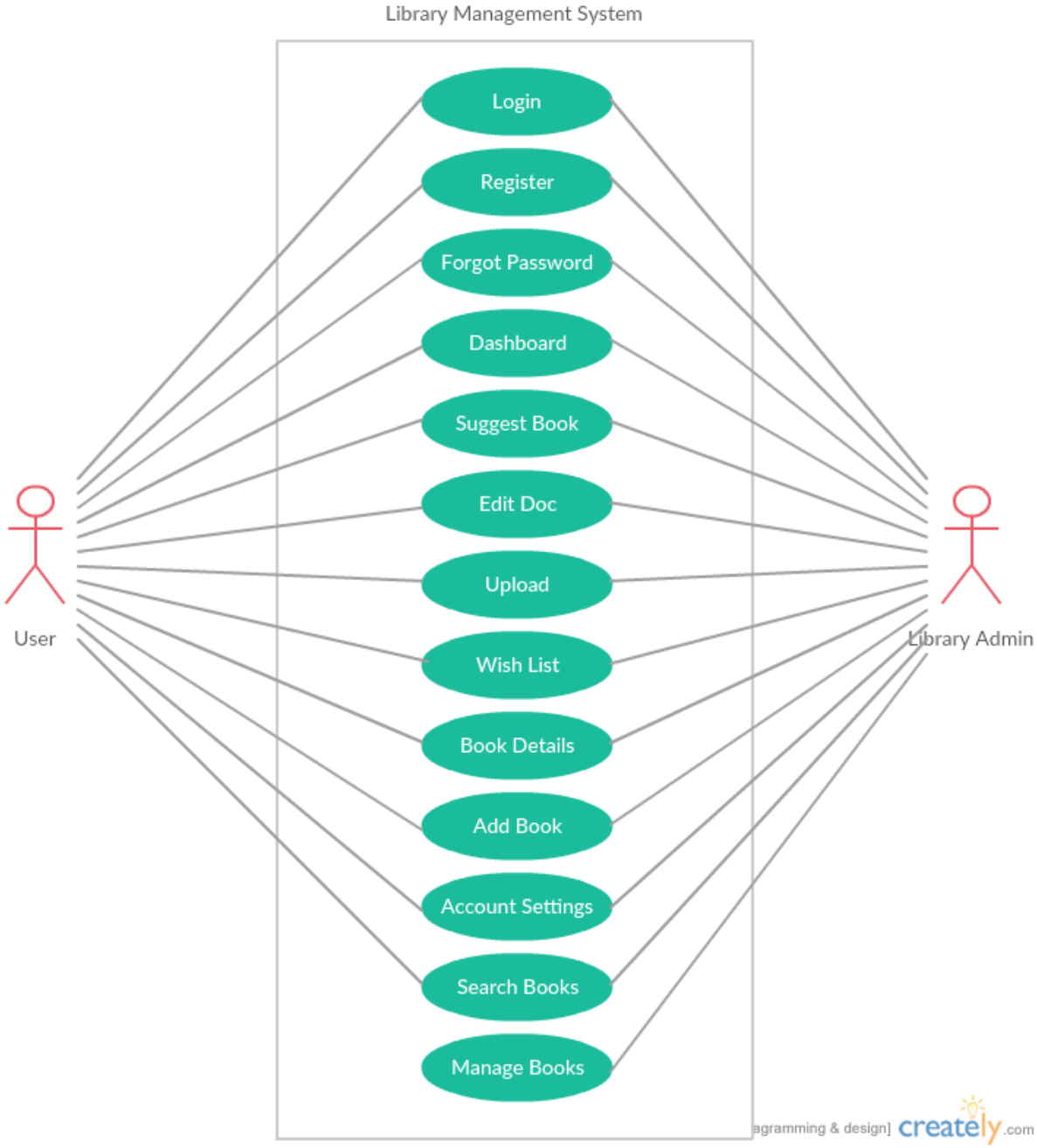
1. **Independence**: Users can work parallel in different parts of the application with minimum dependencies. Application can be independently updated, deployed, maintained on different time schedules. Components can be independently tested.
2. **Security**: Private information can be hidden from other layers.
3. **Reusability**: Components can be reused in different applications
4. **Flexibility**, **Maintainability**, **Scalability** of the application is increased as we separate the presentation layer from Application layer and the application layer in turn from the data access layer.


Presentation Tier
Application Tier
Database Tier

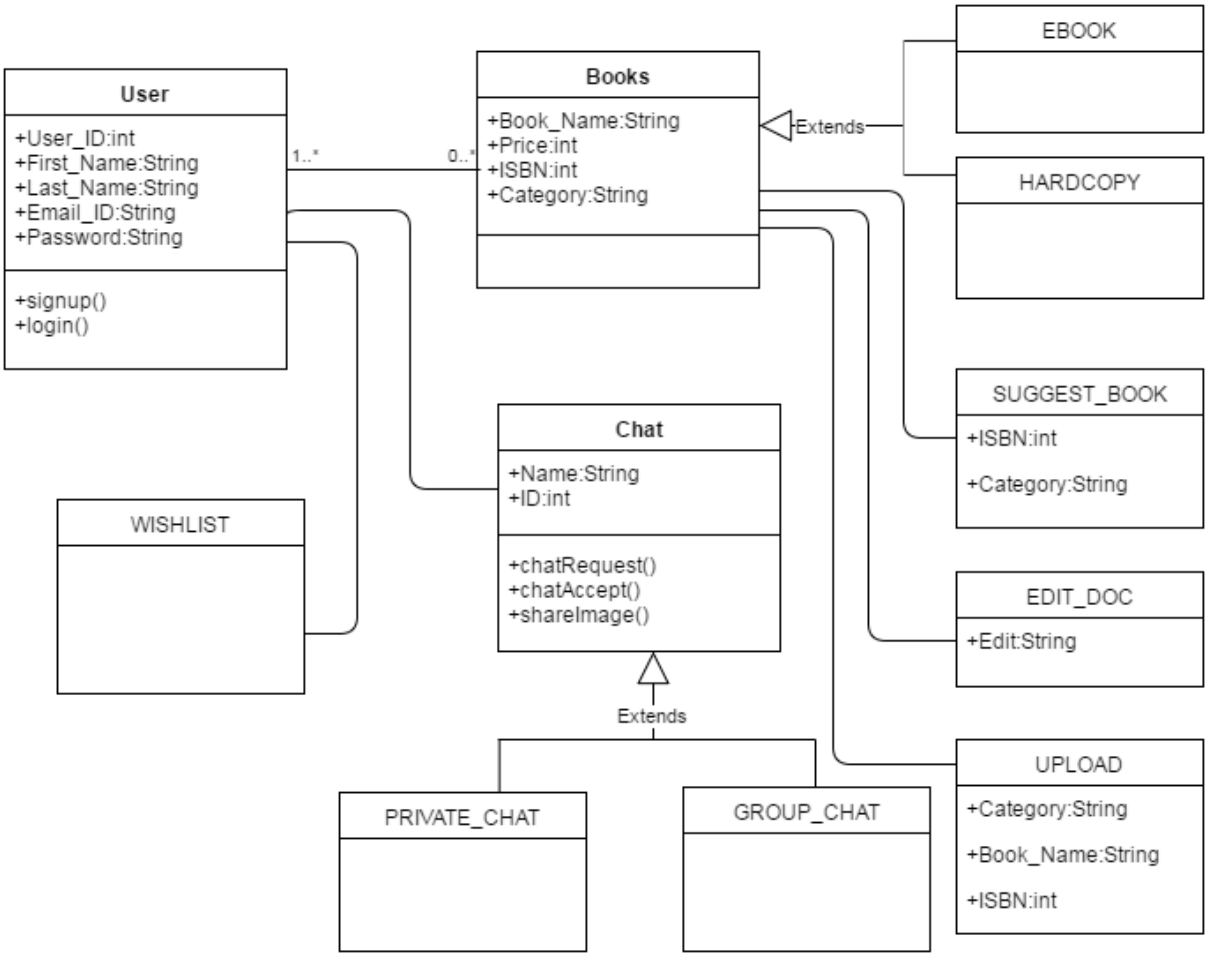
Architecture diagram:



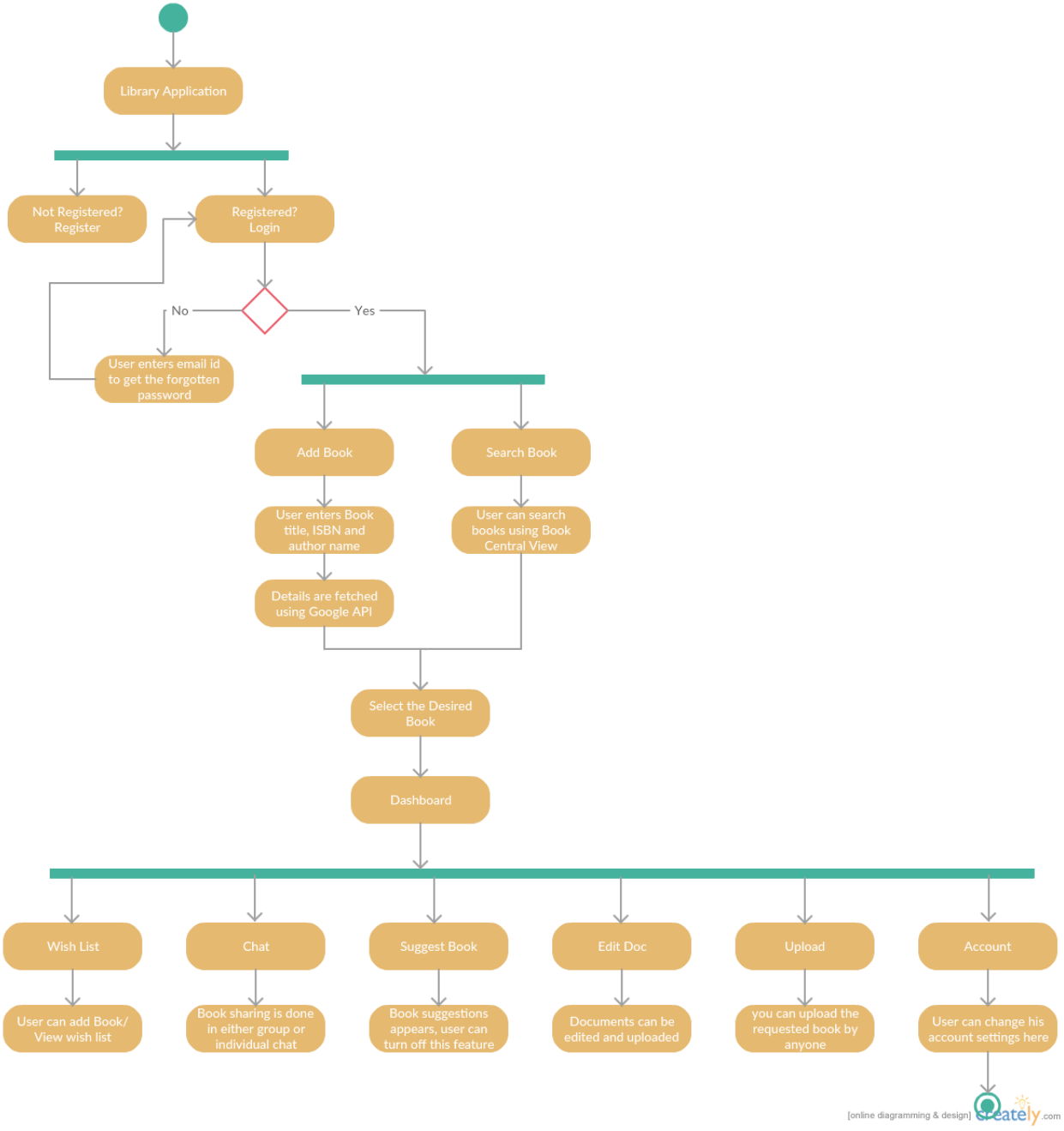
UML Use Case:


Library Management System
Login
Register
Forgot Password
Dashboard
Suggest Book
Edit Doc
Upload
Wish List
Book Details
Add Book
Account Settings
Search Books
Manage Books
User
Library Admin
agramming & design] creately.com

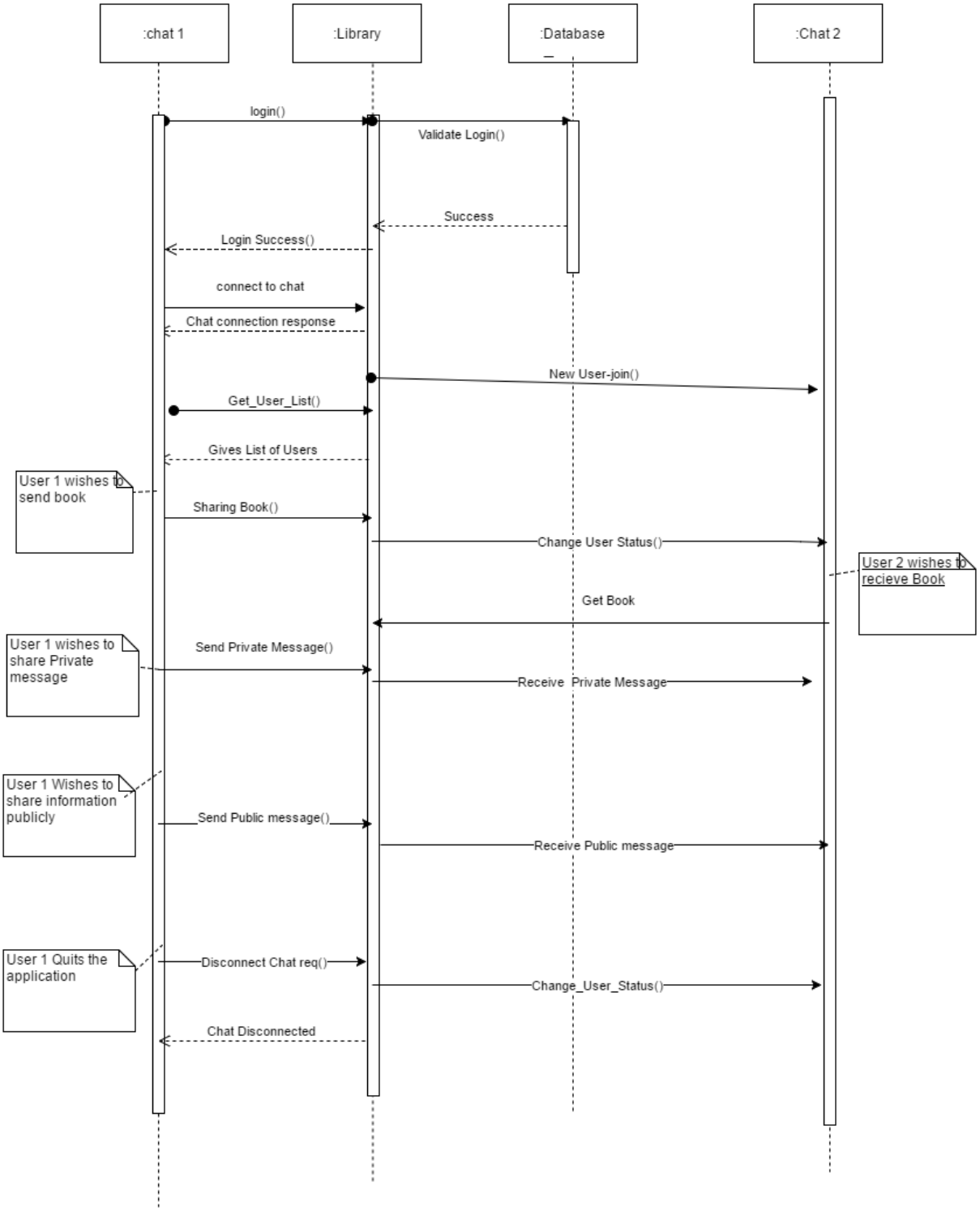
UML Class Diagram:

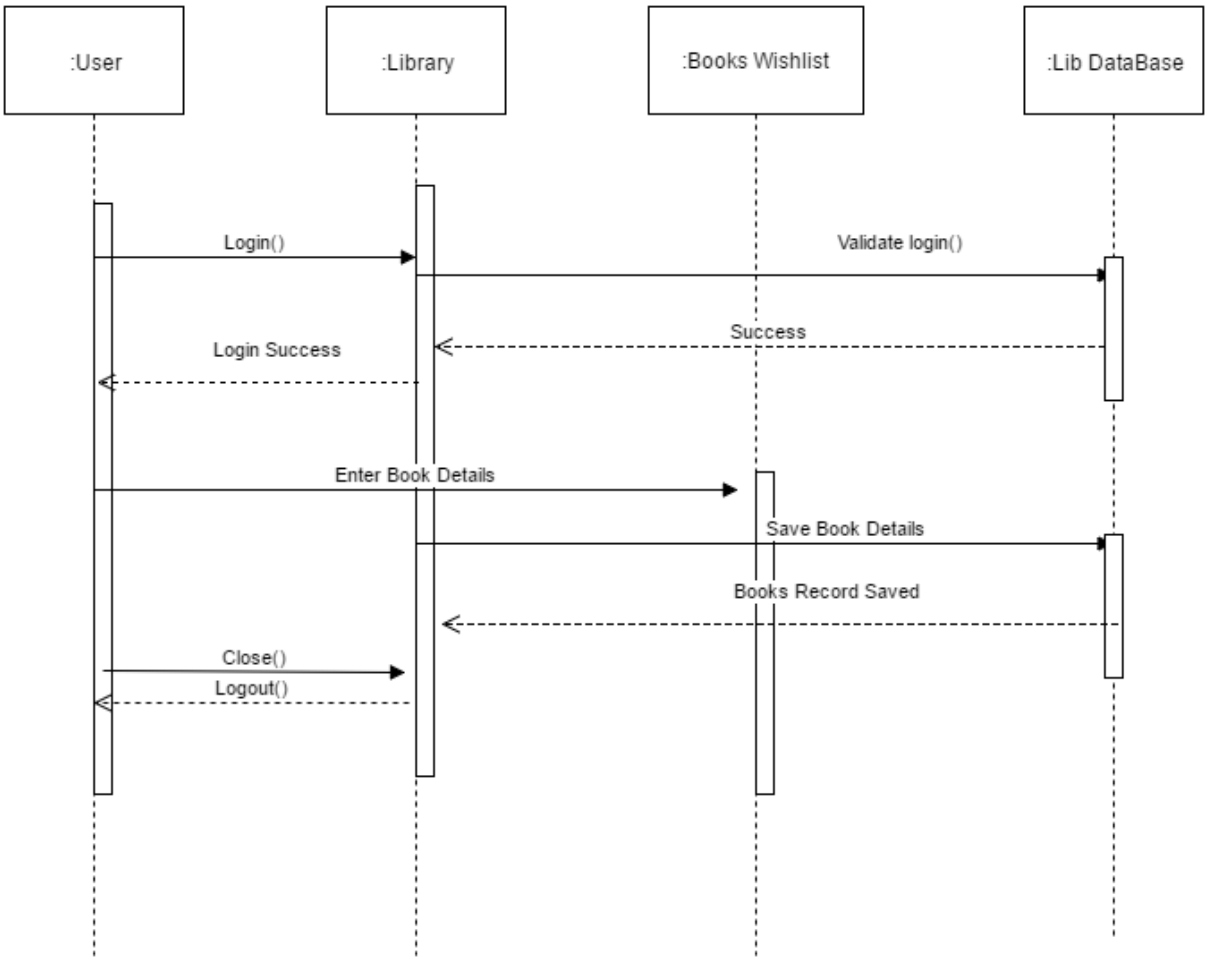


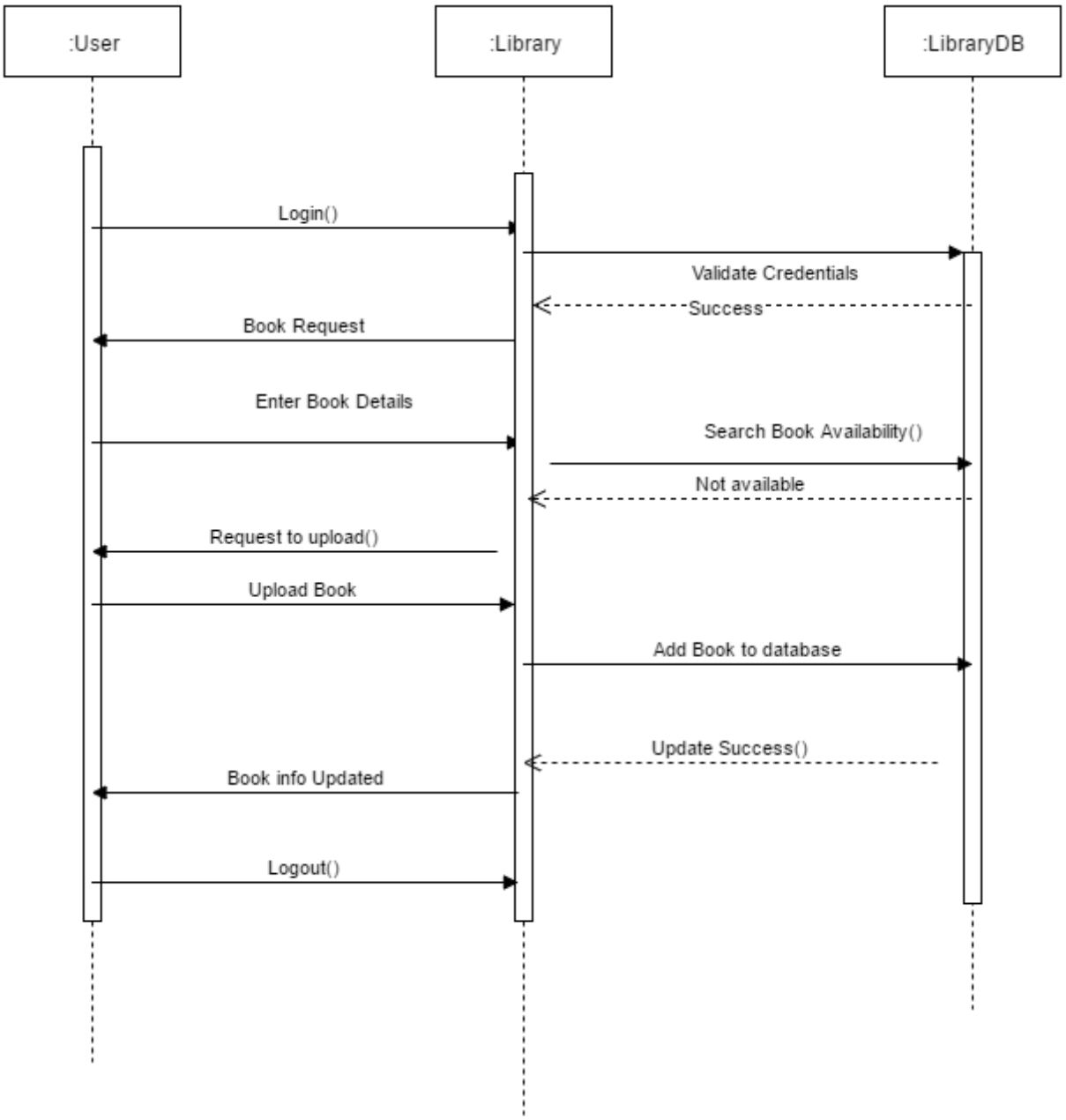
Activity Diagram:



Sequence Diagram:




:User
:Library
:Books Wishlist
:Lib DataBase
Login()
Validate login()
Success
Login Success
Enter Book Details
Save Book Details
Books Record Saved
Close()
Logout()


:User
:Library
:LibraryDB
Login()
Validate Credentials
Success
Book Request
Enter Book Details
Search Book Availability()
Not available
Request to upload()
Upload Book
Add Book to database
Update Success()
Book info Updated
Logout()

Design pattern:

A combination of Observer Pattern and Abstract Factory pattern can be used.

Observer Pattern: Software Design Pattern in which an object, maintains a list of its dependents, called observers, and notifies them automatically of any state changes.

This design pattern is used because; if a user makes any changes like editing a document or uploading a book that change has to be updated by the administrator immediately.

Abstract Factory Pattern: The essence of this pattern is to provide an interface for creating families of related or dependent objects without specifying their concrete classes.