**Work Integrated Learning Programmer**

**M.Tech Software Engineering**



**FULL STACK APPLICATION DEVELOPMENT**

**Submitted by**

**Raghunath Reddy G**

**2023MT93051**

Contents

[1. **GitHub Repositories** (**https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development**) 16](#_Toc182676041)

[**2.** **Demonstration Video** 16](#_Toc182676042)

***Problem statement: BOOK EXCHANGE PLATFORM***

Book lovers frequently accumulate a collection of books they have read and look for other recommendations. They are always eager to explore new reading material. Traditional methods of exchanging books, such as local book swaps or lending among friends, are limited in scope and accessibility. Therefore, it is imperative to have a digital platform that can facilitate book exchanges on a larger scale. This platform should connect users with similar reading interests, enabling them to trade books easily and efficiently. This project aims to develop a full-stack web application that serves as a centralized platform for users to exchange, lend, and borrow books with other users. The platform should provide a user-friendly interface, robust search and recommendation features, and secure transaction capabilities.

***Key Features:***

**User Authentication**: Implement a secure user authentication system to allow users to register, log in, and manage their accounts.

**Book Listing**: Enable users to list books they want to exchange or lend, including details such as title, author, genre, condition, and availability status.

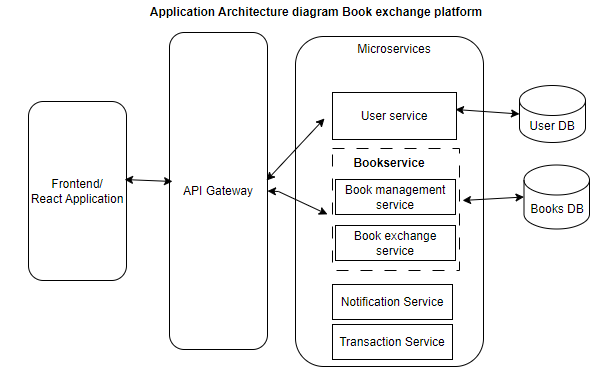
**Book Search:** Provide users with advanced search and filtering options to discover books based on criteria like genre, author, title, location, and availability.

**Exchange Requests**: Allow users to send and receive exchange requests for specific books, including negotiation options for terms such as delivery method and duration.

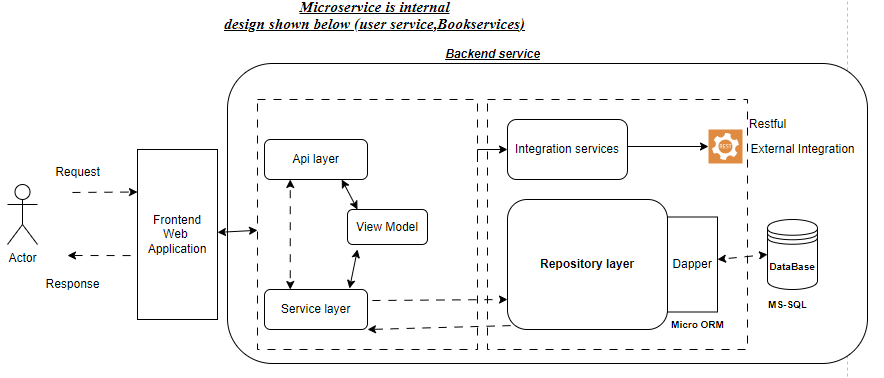
**Messaging System**: Implement a messaging system to facilitate user communication regarding book exchanges, including negotiation details, logistics, and scheduling. (Mock API s can be used)

**User Profiles**: Enable users to create profiles with information about their reading preferences, favorite genres, and books they currently own or wish to acquire.

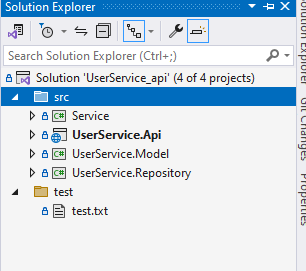
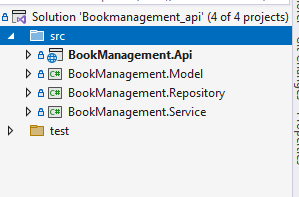
**Transaction Management**: Provide tools for users to track the status of their exchange transactions, including pending requests, accepted exchanges, and completed transactions.

***Architecture design:*** 

***Each Microservice is internal design shown below (user service,Bookservices)***



***Backend api code snippet in n-layer architecture***



Thebackend service is designed using the **microservicce** each micro-service is independently developed & deployed with frequently add the feature in each iteration and deploy to production.

These microserives are *designed based on* ***business domain context & isolated within the bounder***. Each services is communicated via messaging or direct service to service call using restful template.

* **API layer:** will have the Http-Methods like Http GET,POST PUT,DELETE which is exposed as via swagger to consume
* **Service Layer:-** will contain the business logic implemented w.r.t to domain. And also it bind the data from DTO-models to Domain-Data-models.
* **Repository layer**: will have DB connection logic & all DB query and Integrated to Dapper-ORM which will connect to database and load the data to data models.
* **Model layer**: will have the POCO classes or Domain model classes which is replica of the database table which is used to bind the data from table.

***Tech-stack used:***

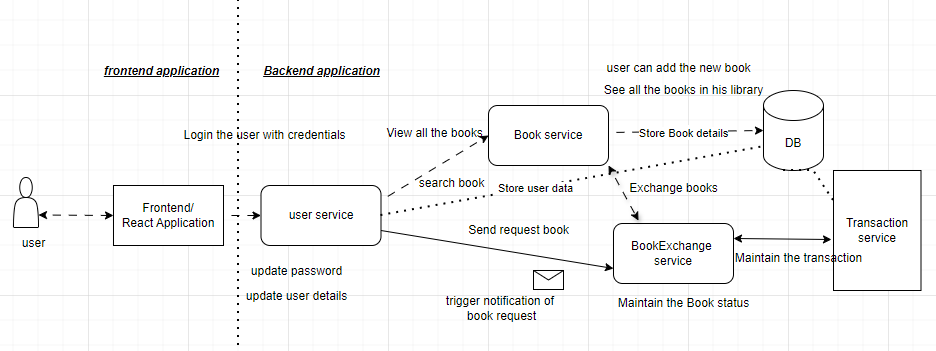
***Back-End stack: -***

* ***.NET core:***  .***Net core restful API*** service is user which is used to build the back business logic and expose it as restful service it will connect to database and pull the data using sql queries.
* ***MS SQL*** : is used as database to store the application user data & book exchange data , this is relational database stores the data in form of rows & coloumns
* ***Dapper***: it is ***MICRO*** ORM which connect the repository & Data model and binds the data from SQL DB to Data model.

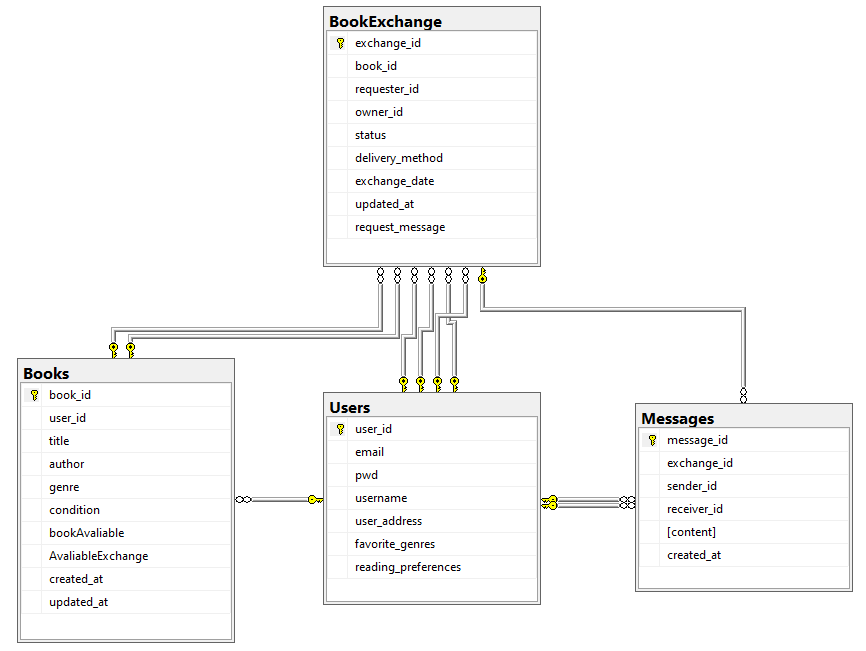
***Front-End stack:***

* ***React Web application: -*** React is used to build the interactive web application with a user friendly. It’s as responsive UI which uses a *boostract .css styling*

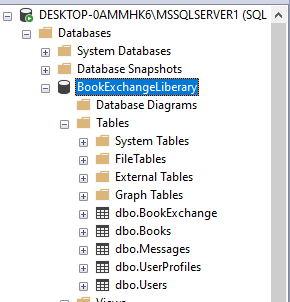
***Flow diagram:***



***Database Design:***



***MS-SQL Database design table details***



***User Story 1: User Authentication***

As a user, I want to securely register, log in, and manage my account, So that I can access and use the book exchange platform.

Acceptance Criteria:

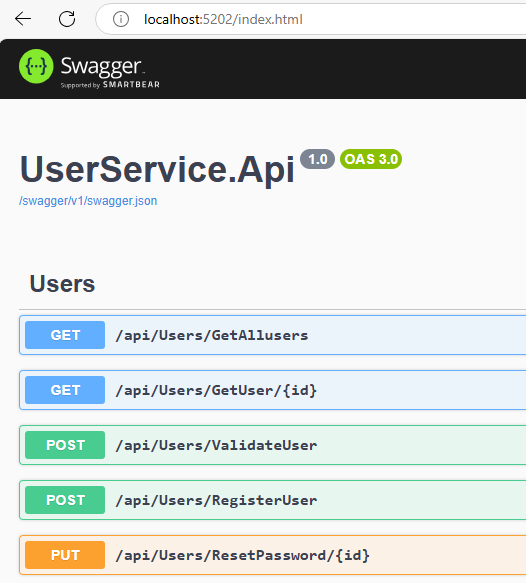
The platform must allow users to register with a valid email and password.

Passwords must be stored securely using encryption.

Users should be able to reset their password via a password recovery system.

Users should be able to log out from their account.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Http Method** | **URI** | **Description** |
| Users | GET | **/api/Users/GetAllusers** | **Get all the users list** |
| GET | **/api/Users/GetAllusers/${userid}** | **Search the User by ID** |
| POST | **/api/Users/RegisterUser** | **Add the new User** |
| PUT | **/api/users/resetPassword/ ${id}** | **Update the user password w.r.t userID** |
| DELETE | **/api/users/${id}** | **Deleted the user** |

***Swagger screen of the “user service”*** 

***User Story 2: Book Listing***

As a user, I want to list books that I want to exchange or lend, So that others can browse and request the books I offer.

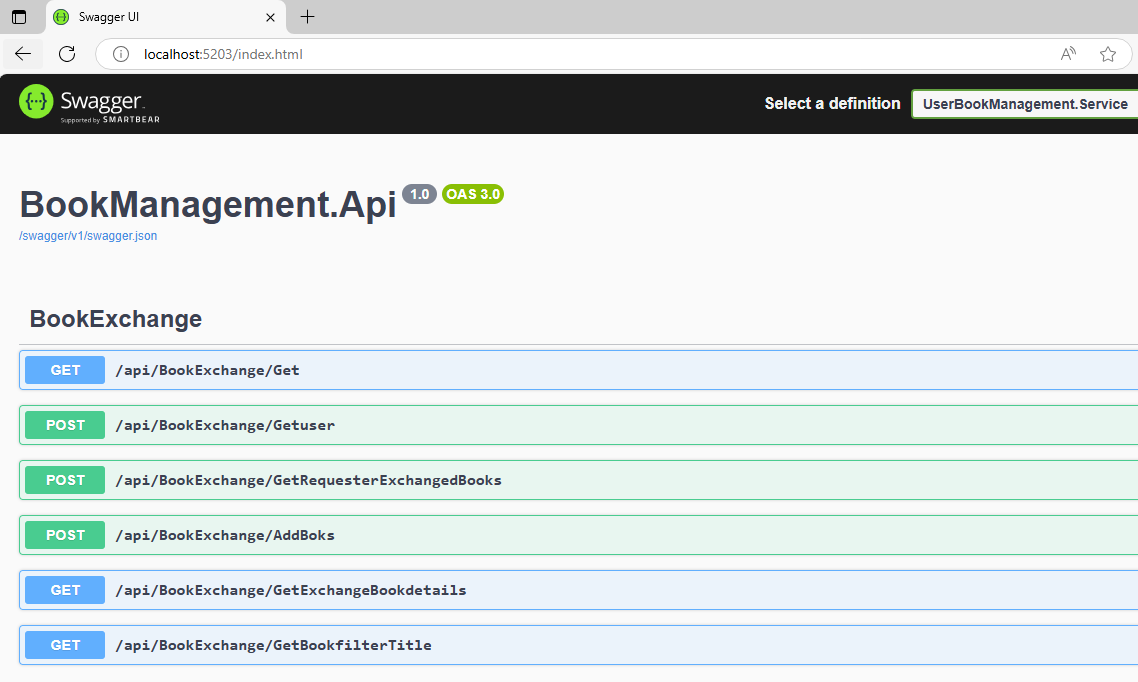
Acceptance Criteria:

Users should be able to add a book to their list by providing details such as title, author, genre, condition, and availability status.

Each book listing must have a unique ID associated with a user’s profile.

Users should be able to edit or delete book listings at any time.

The book listing must be displayed in the user's profile and searchable by others.



|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Http Method** | **URI** | **Description** |
| ExchangeBook  Service | GET | **/api/BookEchange/Get** | **Get all the Books for Exchange** |
| POST | **/api/BookEchange/GetUser** | **Search the Books available w.r.t UserID** |
| POST | **/api/ BookEchange /AddBooks** | **Add the New books for available for Exchange** |
| GET | /api/ BookEchange /GetBookFilterByTitle | **Search the Books available for Exchange w.r.t Title** |
| POST | **/api/ BookEchange /GetRequestExchangeBooks** | **Raise a request for books exchange** |

**User Story 3: Book Search**

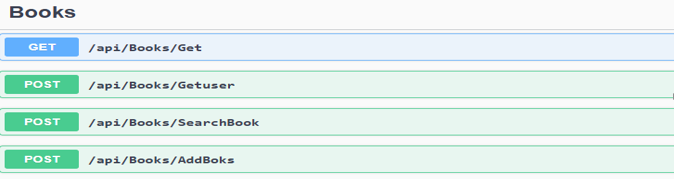
As a user, I want to search for books based on criteria such as title, author, genre, and location,

Acceptance Criteria:

The platform must provide a search bar where users can enter keywords like title, author, or genre.

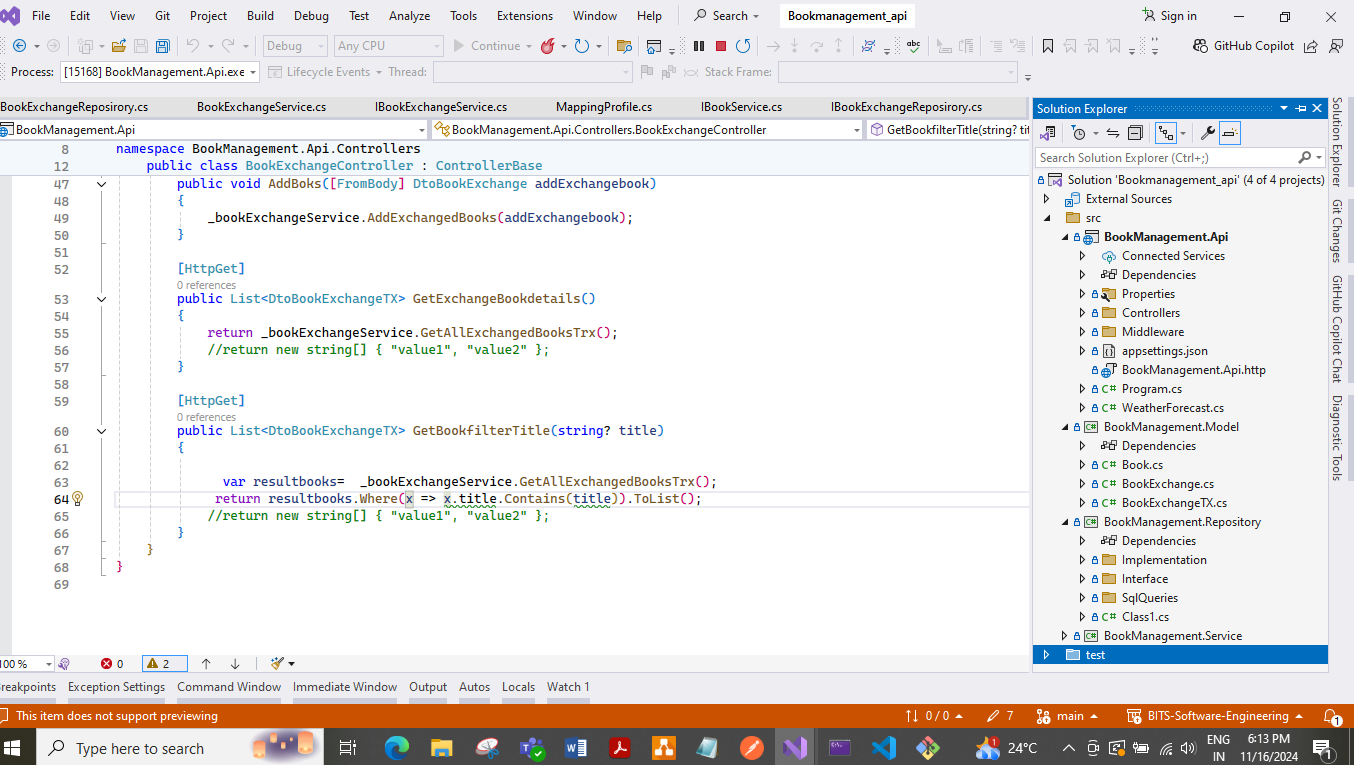
The platform should allow users to filter search results by availability status, genre, and location.

Users must be able to view detailed information about a book (title, author, condition, etc.) when clicking on a search result. The search results should be paginated or load incrementally to handle large datasets.

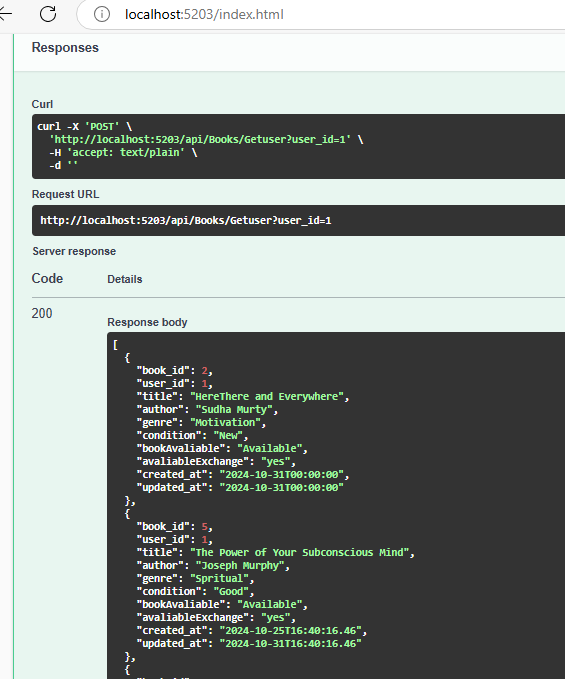


|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Http Method** | **URI** | **Description** |
| Books  Service | GET | **/api/Books/Get** | **Get all the Books in library /store** |
| POST | **/api/ Books /GetUser** | **Search the Books available w.r.t UserID** |
| POST | **/api/ Books /SearchBook** | **Search book w.r.t title , gener,** author |
| POST | **/api/ Books /AddBooks** | **Add the book in this list** |

***Backend code structure***

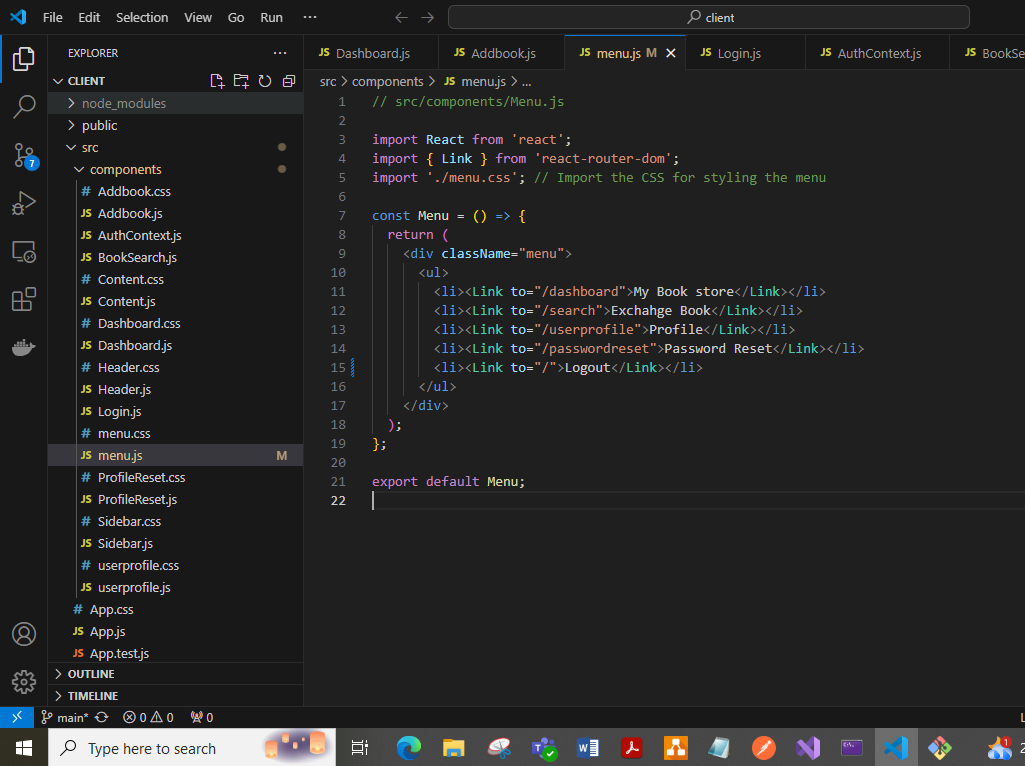


***Below are the API books response example***

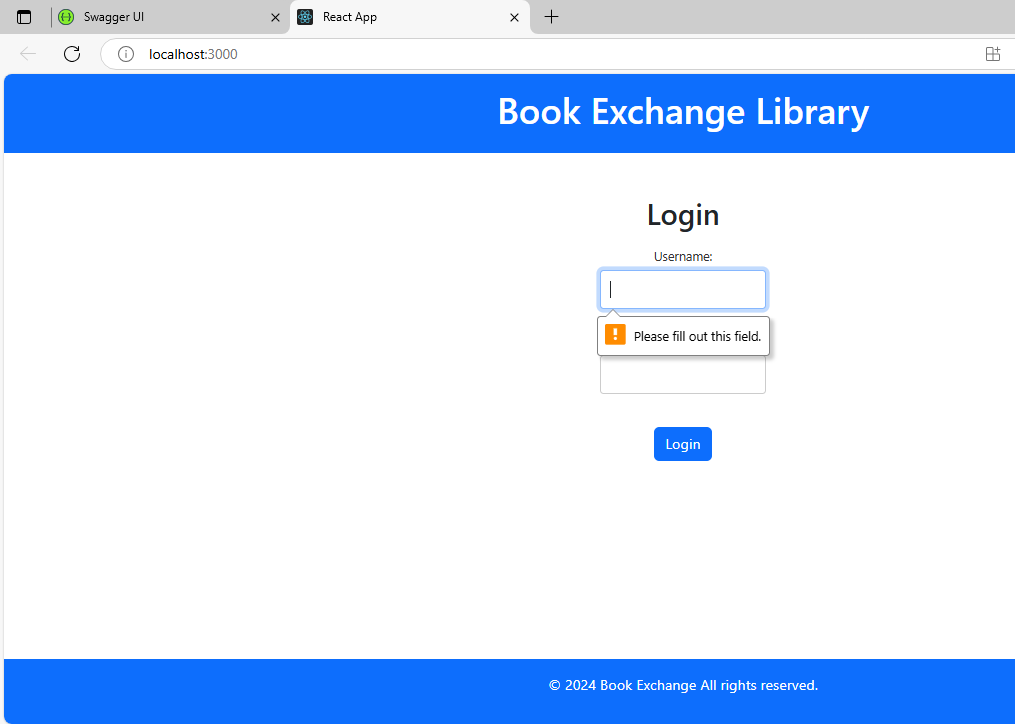


***Frontend React application screen shot:-***

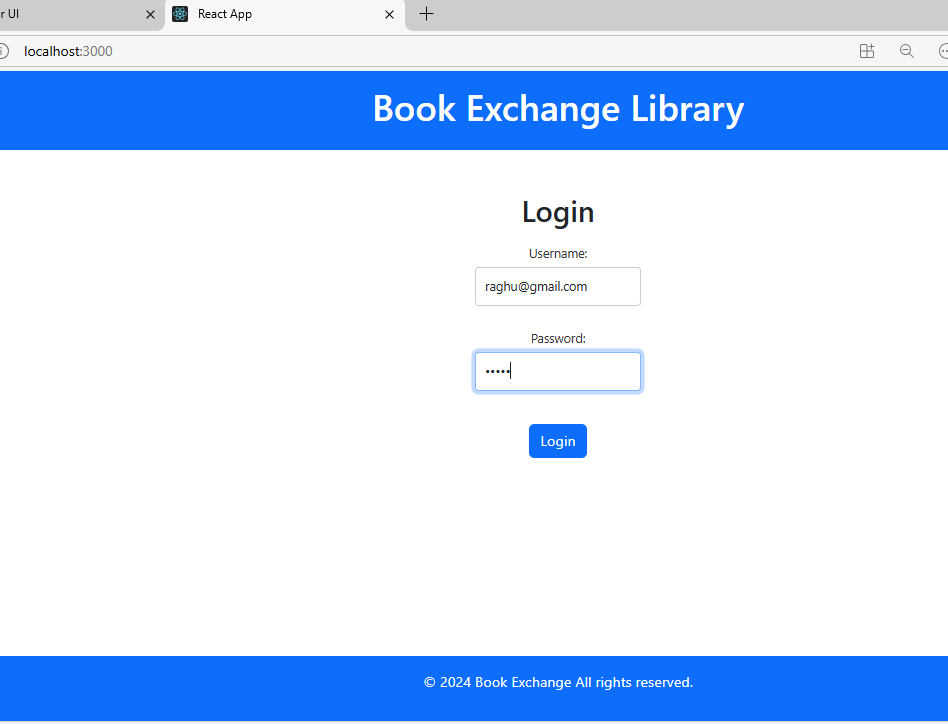
***Code structure:***



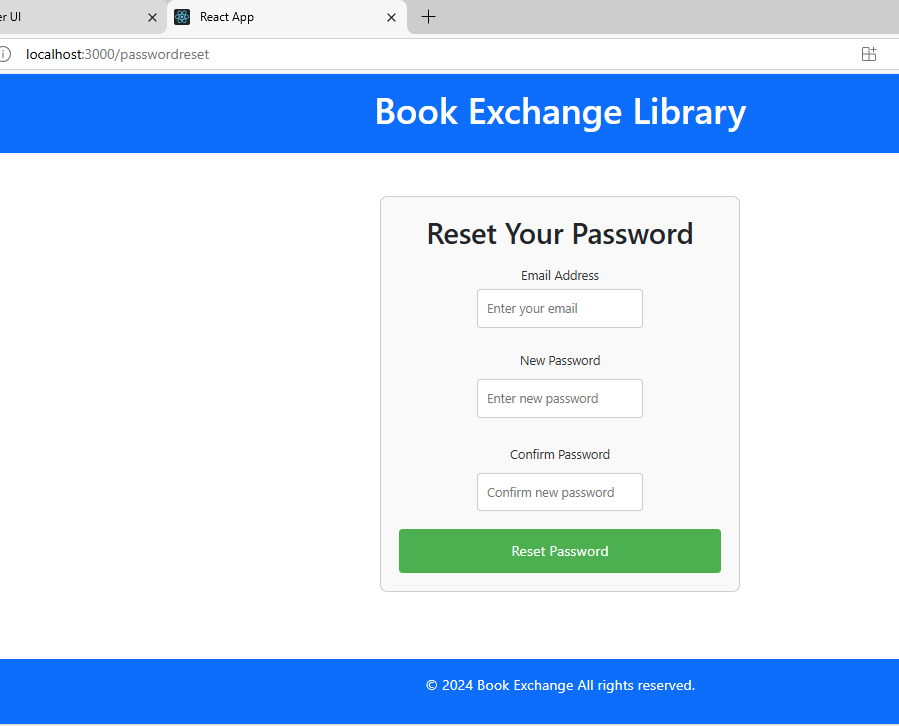
***Login screen:- login with validation***



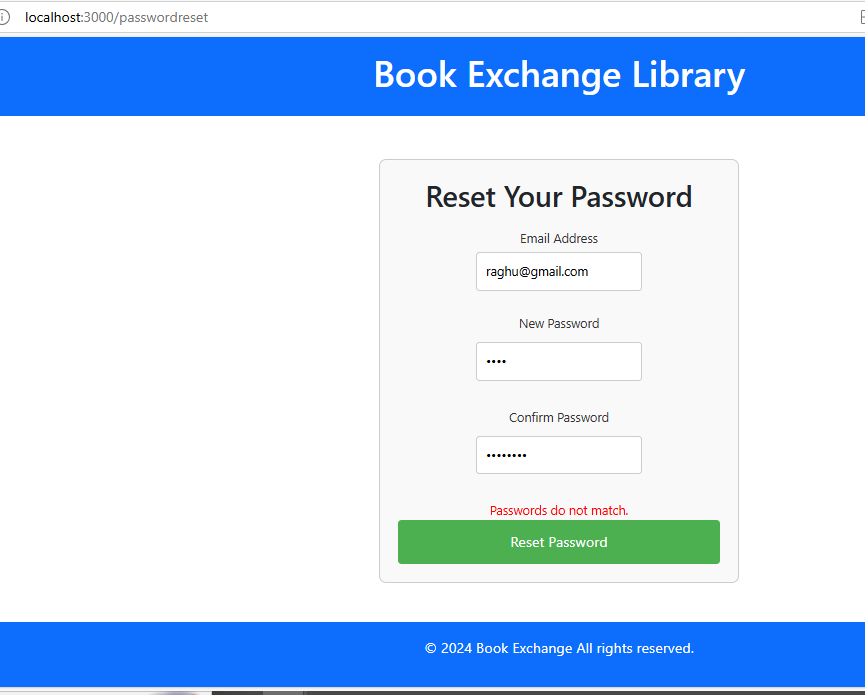
***Login screen: - login with username & password***



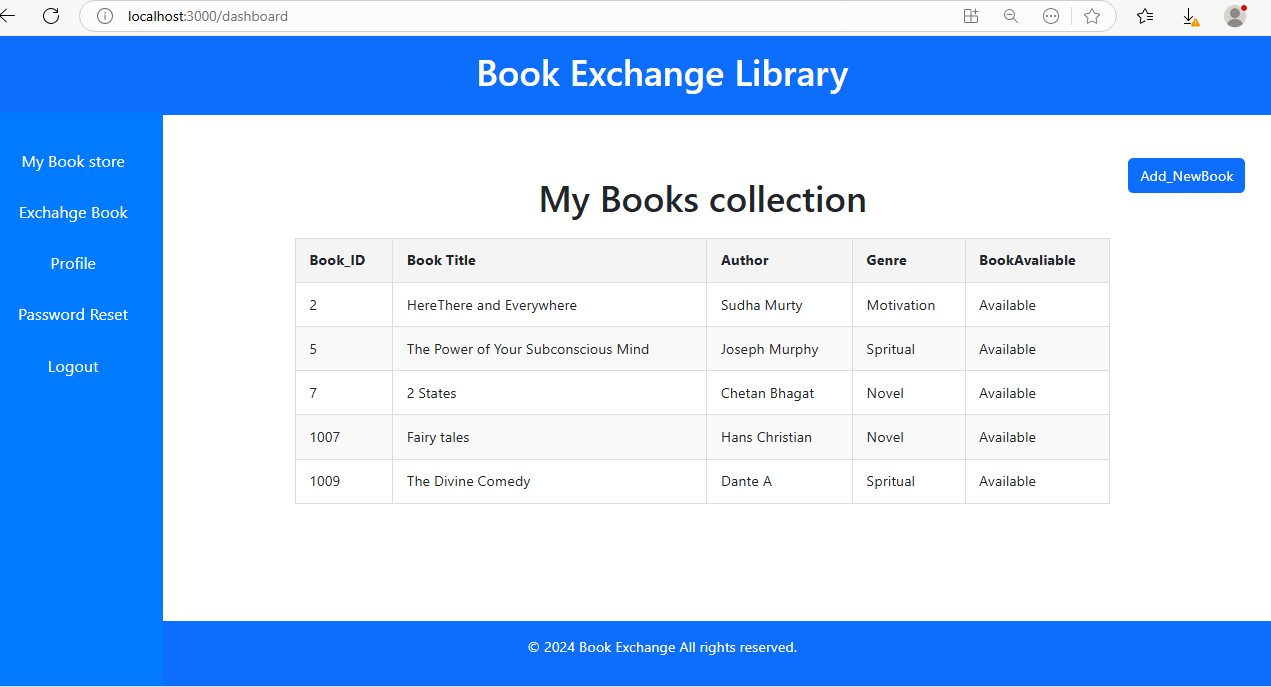
***Password Reset:***



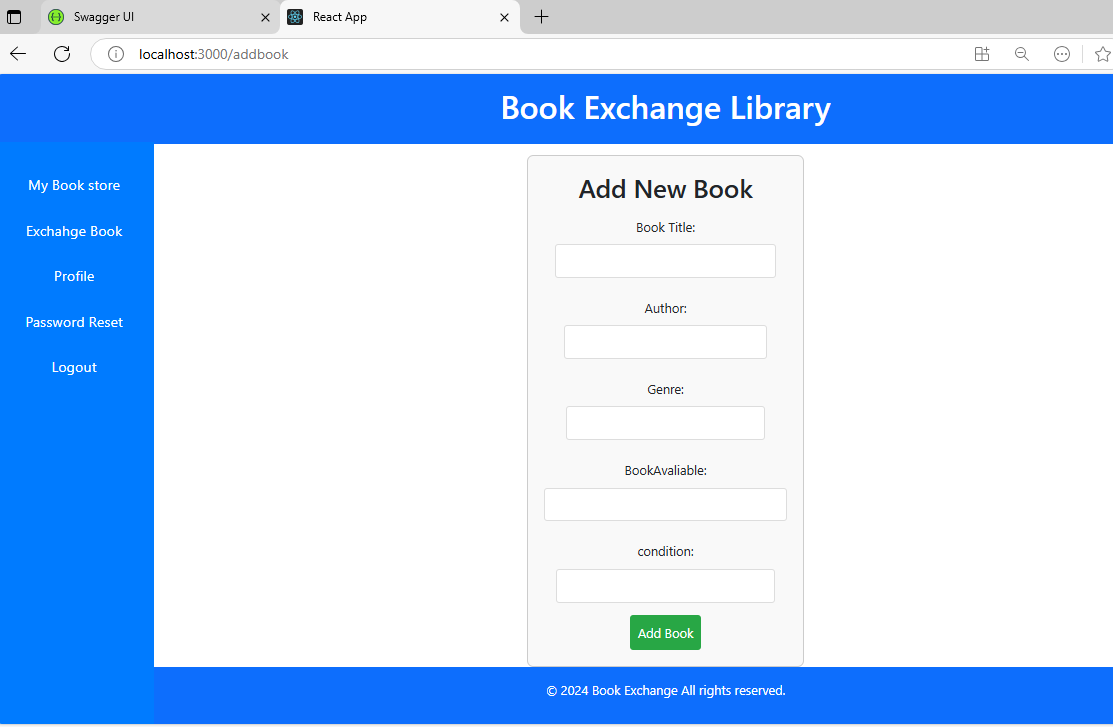
***Password Reset: with validation***



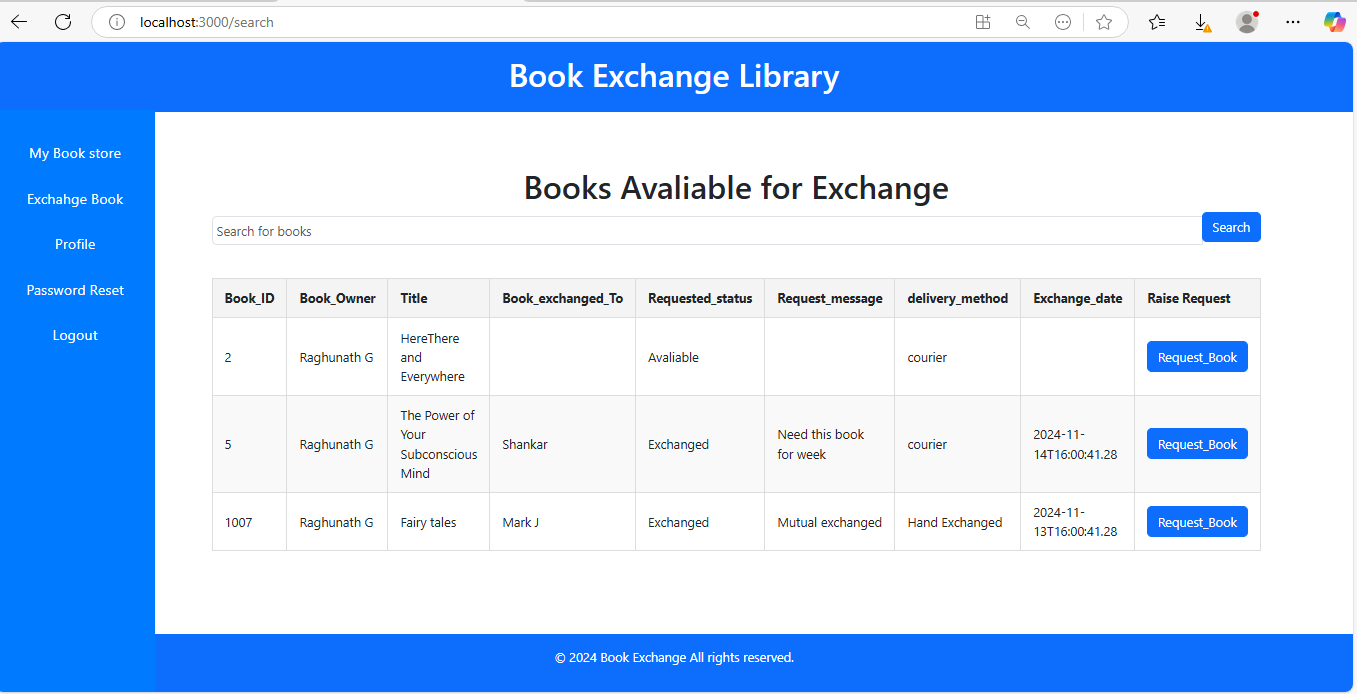
***Dashboard: once login it will show all the books collection in his library/store***



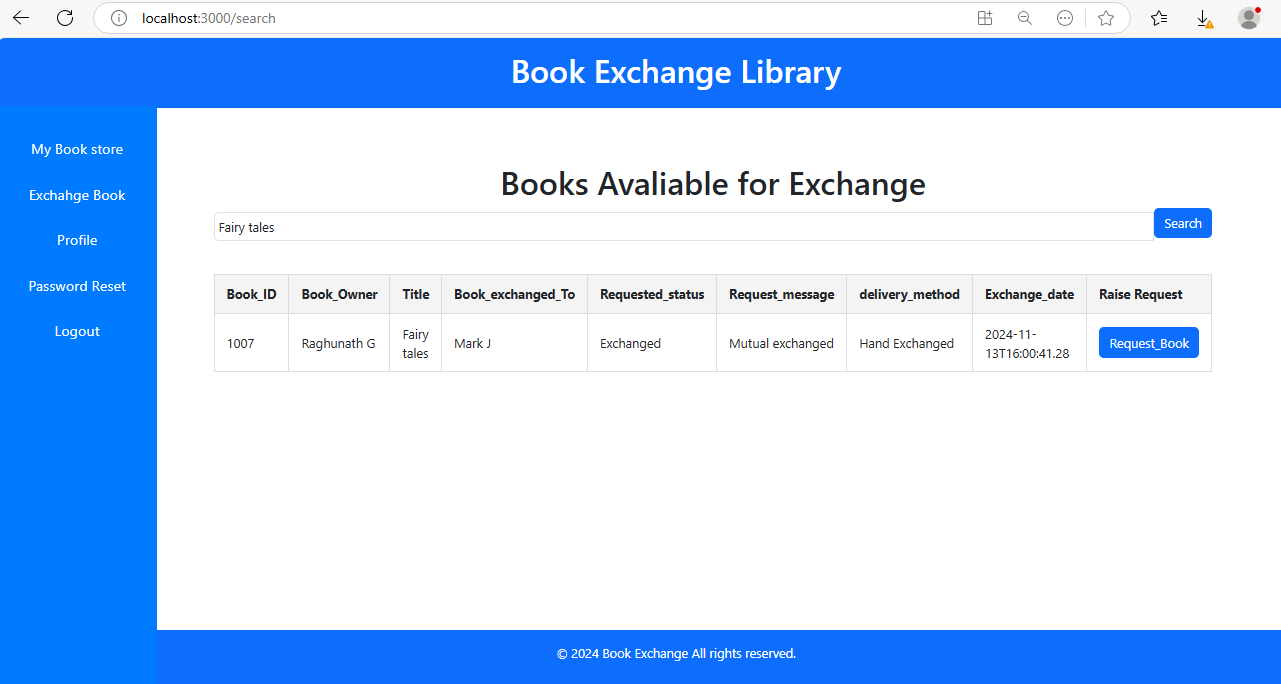
***Add the new book in the collection list:***



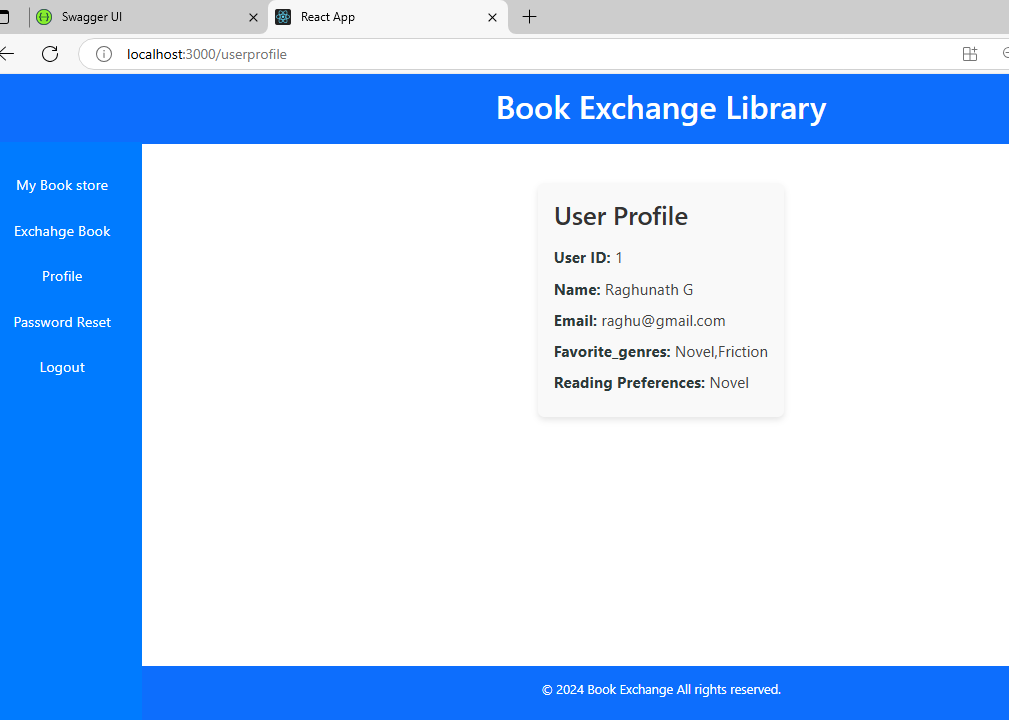
***List all the Books available for Exchanges with other***



***Search the book with title shown below:***



***LogedIn user profile:***



# **GitHub Repositories** ([**https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development**](https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development))

**Backend services:-**

* ***UserService API*** *:*-

[***https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development/UserService\_api***](https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development/UserService_api)

* ***BookService API:-***

[***https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development/Bookmanagement\_api***](https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development/Bookmanagement_api)

**Frontend React application:-**

* ***React Application:-***[***https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development/frontend/book-exchange-liberary/client***](https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development/frontend/book-exchange-liberary/client)

# **Demonstration Video**

1. **Documentation & Architecture diagram:-**

* [***https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development/document***](https://github.com/raghunathreddy/BITS-Software-Engineering/tree/main/3rdsem/fullstack-development/document)