

BookSwap Hub

Saurabh Kale

IFT 458/554: Middleware Programming & Database Security

Dinesh Sthapit

December 1st, 2023

Table of Contents

Abstract	3
Introduction	3
Problem Statement	4
Scope and Limitations	4
Justification of Technology Stack for Bookswap Hub	4
Project Design	5
Screenshots	10
Testing	21
Implementation	25
Learning Outcomes	27
Constructive Feedback	27
References	28

Abstract

BookSwap Hub is an innovative online platform designed to facilitate the exchange and sharing of books among avid readers. The platform serves as a dynamic hub where users can add, discover, and exchange books with fellow enthusiasts. With a user-friendly interface, members can easily list their books for borrowing or trading, fostering a vibrant community centered around a shared love for literature. BookSwap Hub not only provides a seamless experience for adding and managing books but also incorporates features for requesting and accepting exchanges, enhancing the overall reading experience. Join our literary exchange network and unlock a world of diverse stories, connecting readers and fostering a sense of community through the joy of sharing books.

Introduction

Bookswap Hub is a web application designed to facilitate book exchanges among users, allowing them to add, browse, and request books for borrowing or trading. The project employs a Node.js backend with Express for routing and MongoDB as the database, while the frontend is rendered using EJS templates. User authentication is implemented using JWT tokens, and the system includes features such as user registration, login, book addition, and request processing. The application incorporates various middleware functions for authentication and authorization, ensuring secure access to specific routes. The codebase follows the Model-View-Controller (MVC) architecture, with separate controllers handling user authentication, book-related operations, and middleware managing authentication and authorization. The project includes routes for adding, updating, and deleting books, as well as handling book requests and user authentication. Additionally, there are features for accepting book requests and an about page providing information about the student creator. Overall, Bookswap Hub is a comprehensive book exchange platform with a focus on user-friendly interactions and secure functionalities.

Problem Statement

Bookswap Hub aims to address the need for a user-friendly and secure platform for book enthusiasts to exchange their books through borrowing or trading. The existing problem lies in the lack of a dedicated application that seamlessly connects users with similar literary interests, making the process of book exchange cumbersome and less efficient. Bookswap Hub seeks to streamline this experience by providing a centralized hub where users can easily add, browse, and request books, enhancing the overall accessibility and convenience of book exchanges while ensuring the security of user data and transactions.

Scope and Limitations

The scope of Bookswap Hub encompasses creating an intuitive and efficient platform for users to facilitate book exchanges, fostering a community of avid readers. Users can seamlessly add, explore, and request books for borrowing or trading, enhancing the overall accessibility of diverse literary collections. The platform's scope extends to user authentication, authorization, and transactional security to safeguard sensitive information. However, certain limitations may include the platform's reliance on user-generated content for book listings, potentially leading to variations in data quality. Additionally, real-time communication features for users to discuss exchange details are not implemented, and the platform does not currently support international book exchanges. Continuous improvement and user feedback will be integral to addressing these limitations and refining the Bookswap Hub experience.

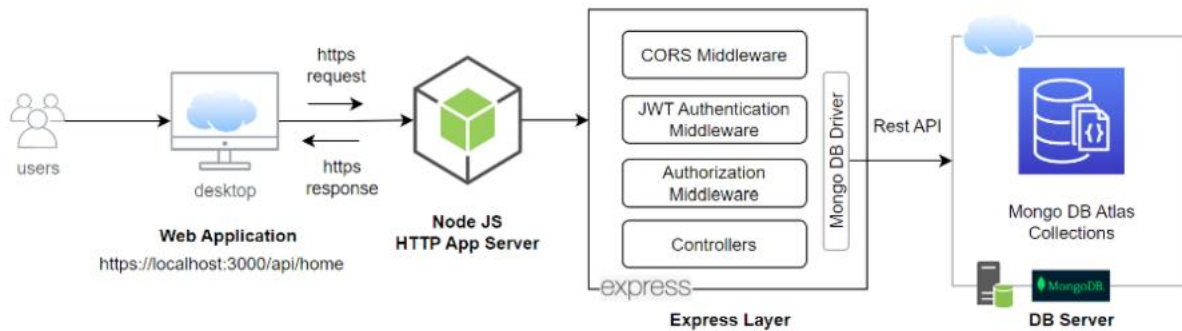
Justification of Technology Stack for Bookswap Hub

1. **Node.js:**
 - Non-blocking, event-driven architecture ensures efficient handling of concurrent connections.
 - Suitable for real-time applications, enhancing responsiveness.
2. **Express.js:**
 - Lightweight and flexible framework simplifies server-side development.
 - Streamlines routing, middleware integration, and overall application structure.
3. **MongoDB:**
 - NoSQL database accommodates the dynamic and schema-less nature of book-related data.
 - Provides scalability and seamless integration with Node.js.
4. **JWT (JSON Web Tokens):**
 - Enhances security by securely transmitting user information between client and server.
 - Supports stateless authentication, improving scalability and performance.
5. **EJS Templates:**

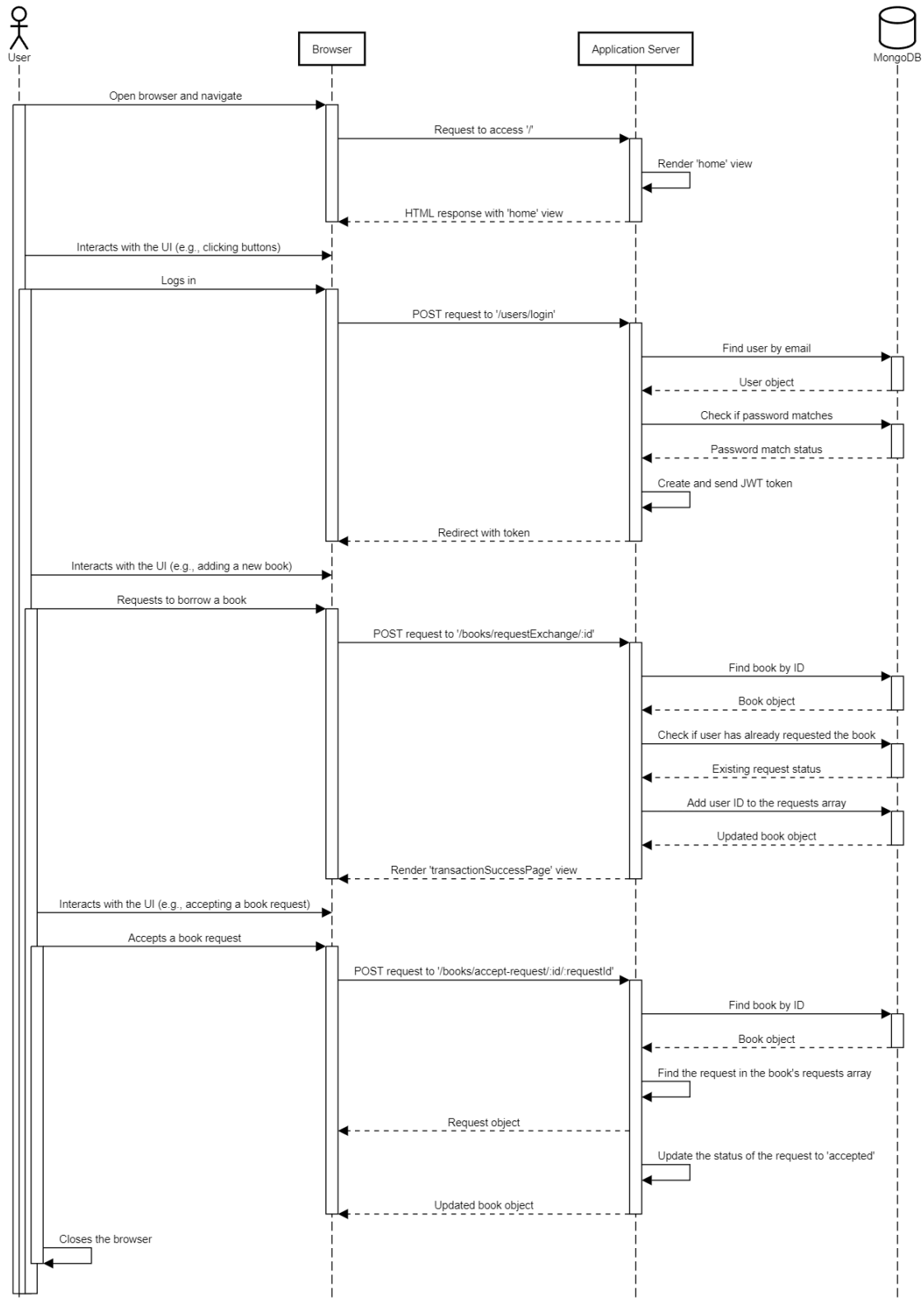
- Embedded JavaScript templates facilitate dynamic HTML rendering on the client side.
 - Integrates seamlessly with Express.js for efficient server-side rendering.
6. **Bootstrap:**
- Ensures a responsive and visually appealing user interface.
 - Accelerates front-end development with pre-designed UI components.

Project Design

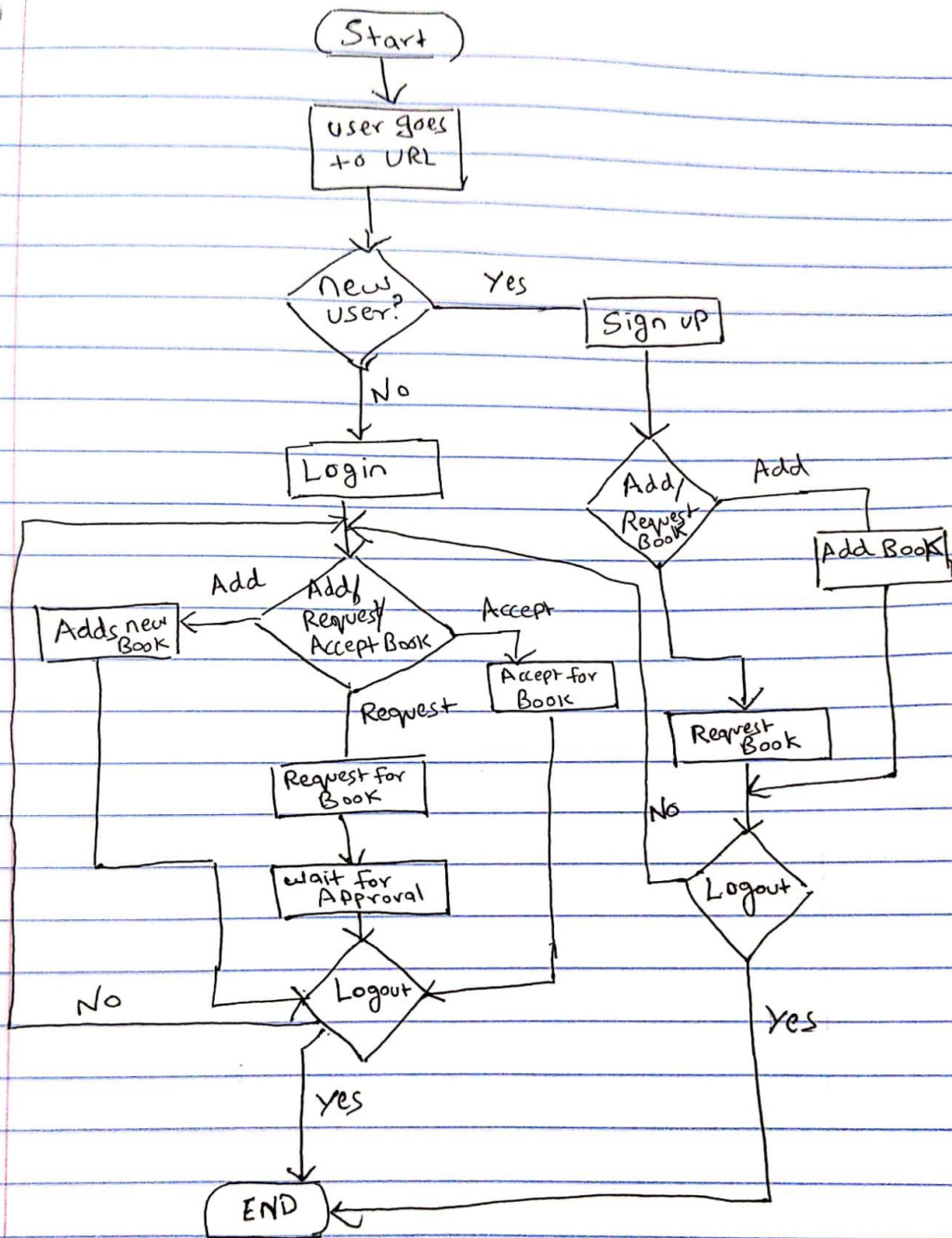
Architecture and Design



Sequence Diagram-



Flowchart



Pseudocode-

// User Actions

// Login

function userLogin(email, password):

 if email and password are present:

 send login request to server

 if login successful:

 store user token locally

 redirect to user dashboard

 else:

 display login error message

 else:

 display error, missing credentials

// Add a Book

function addBook(title, author, description, exchangeType):

 if user is authenticated:

 send add book request to server

 if book added successfully:

 redirect to user's book list

 else:

 display error, book not added

 else:

 redirect to login

// Request a Book

function requestBook(bookId):

 if user is authenticated:

 send book request to server

 if request successful:

 display success message

 else:

 display error, request failed

 else:

 redirect to login

// Accept Book Request

function acceptBookRequest(bookId, requestId):

if user is authenticated and is the owner of the book:

 send accept request to server

 if acceptance successful:

 display success message

 else:

 display error, acceptance failed

else:

 redirect to login

// Logout

function userLogout():

 clear locally stored user token

 redirect to home page

// Usage Examples

userLogin("test@gmail.com", "test12345")

addBook("Wings of Fire", "APJ Kalam", "Autobiography", "trade")

requestBook("12345") // Assuming '12345' is the book ID

acceptBookRequest("67890", "54321") // Assuming '67890' is the book ID and '54321' is the request ID

userLogout()

API Specifications-

Authentication:

Action	Endpoint	Method	Payload
Signup	/users/signup	POST	name,email, password, passwordConfirmation
Login	/users/login	POST	Email, Password
Logout	/users/logout	GET	-
About	/users/about	GET	-

Books:

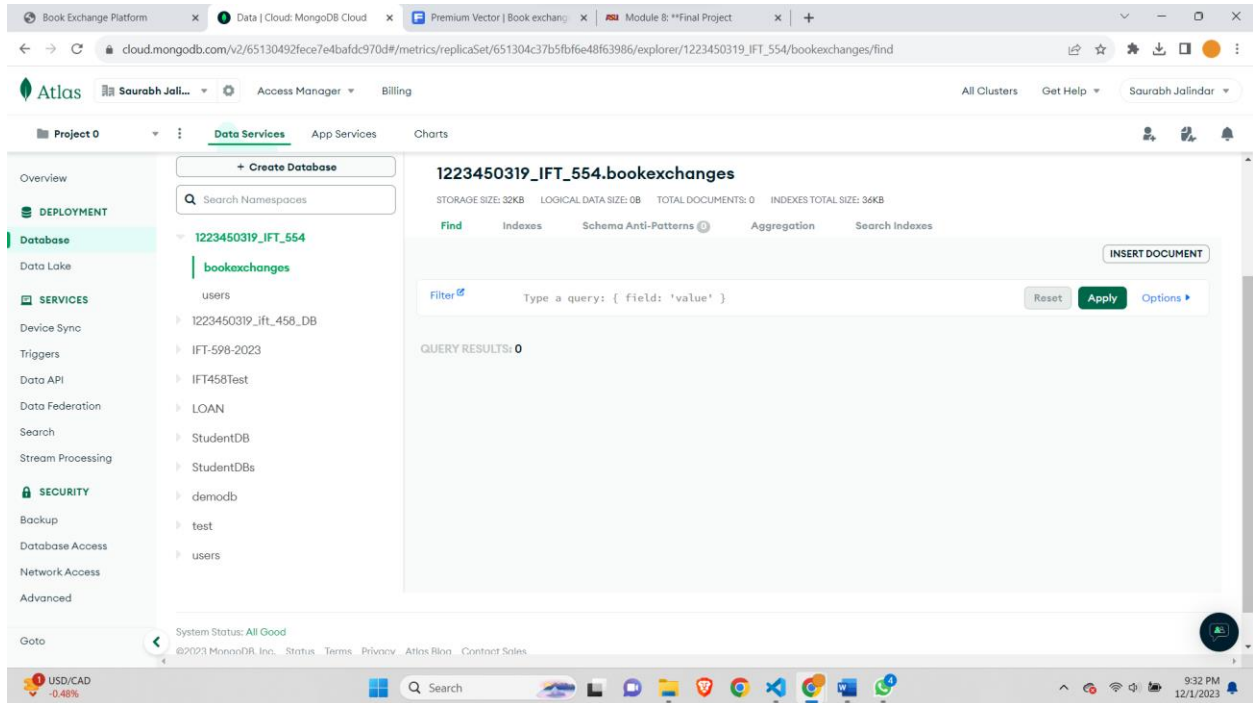
Action	Endpoint	Method	Payload
Add Book	/book/createBook	POST	Title,author,description, exchangeType
Get Books	/books/getBooks	GET	-
Request Book	/books/requestExchange/:id	POST	Id (Book Id)
Accept Request	/books/accept-request/:id/:requestId	POST	Id (book Id), request Id

Screenshots

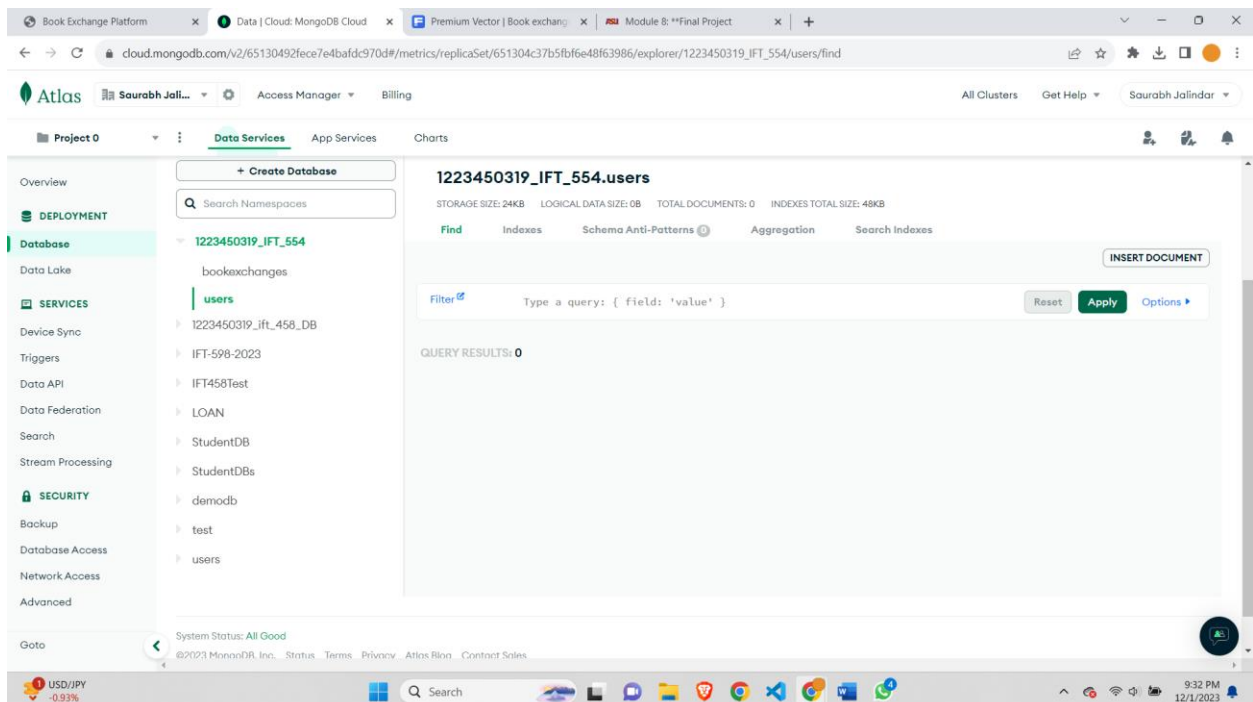
Database-

Created New Database

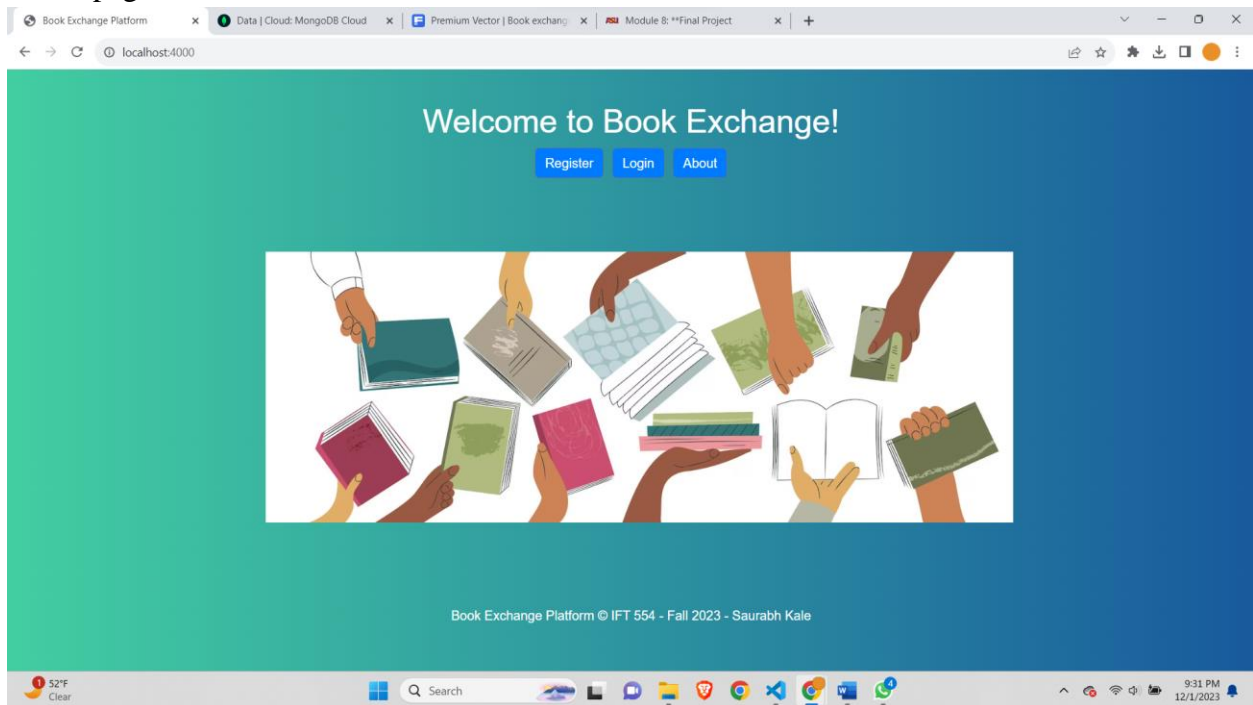
Bookexchanges-



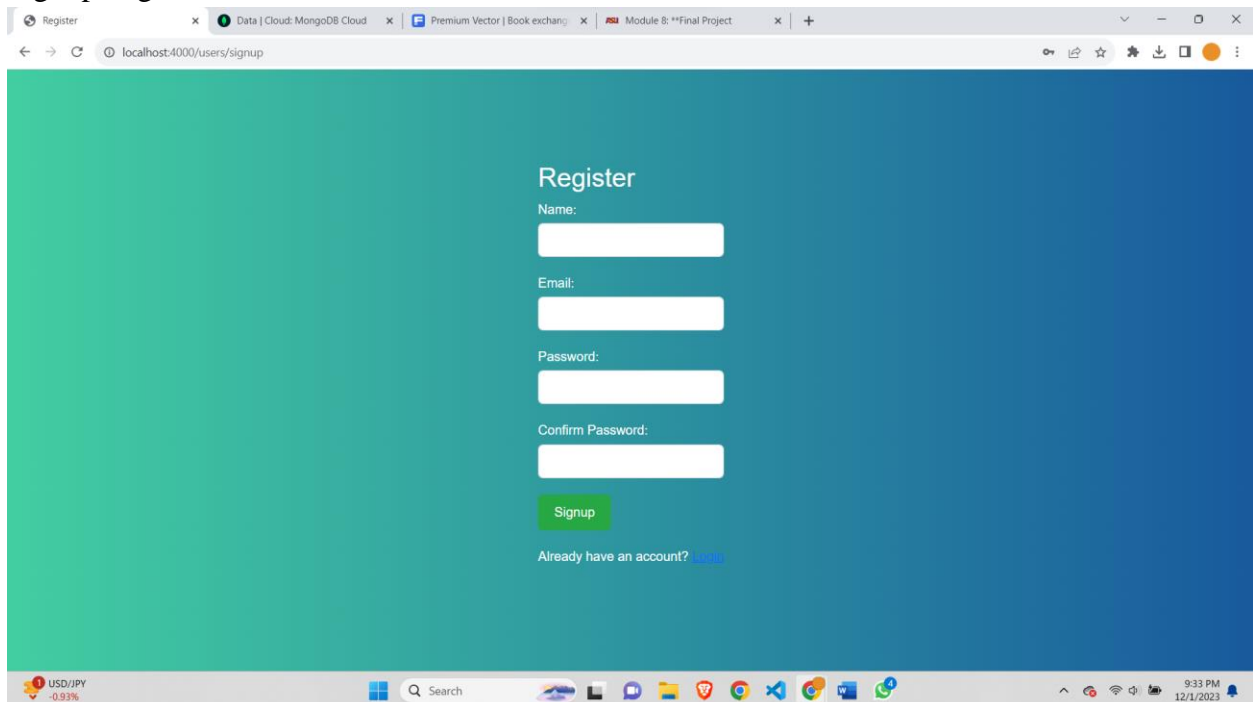
User-



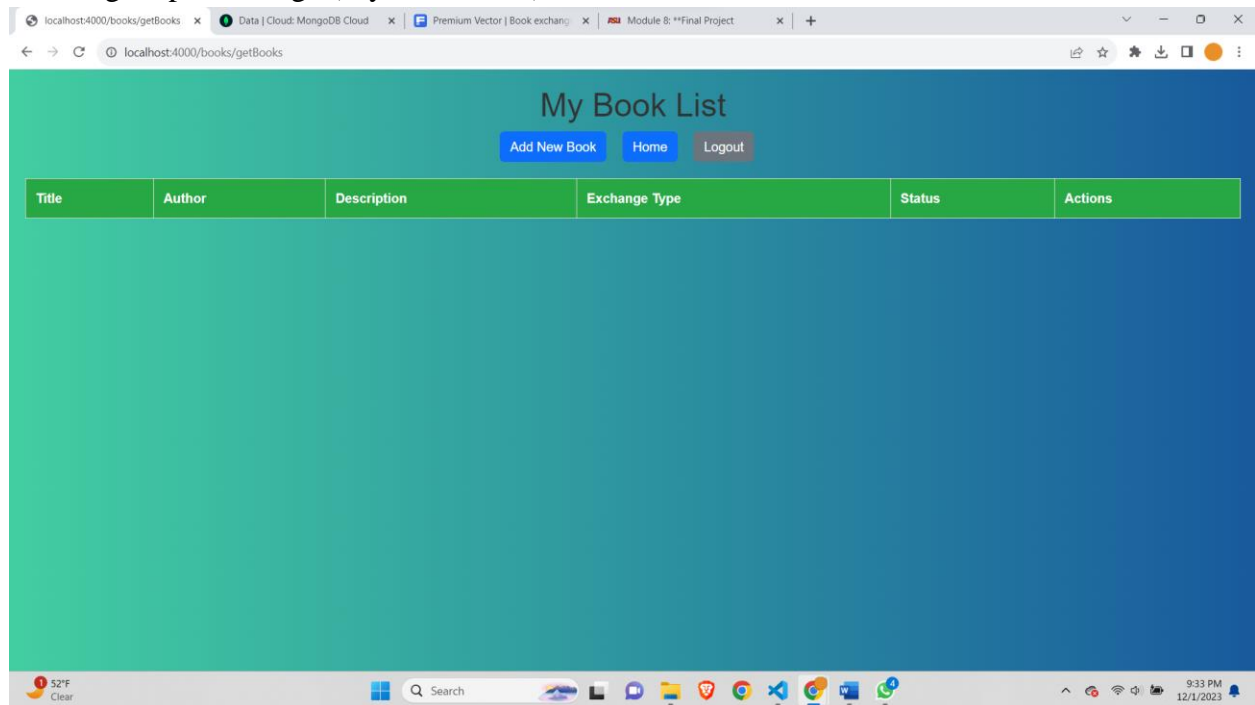
Home page-



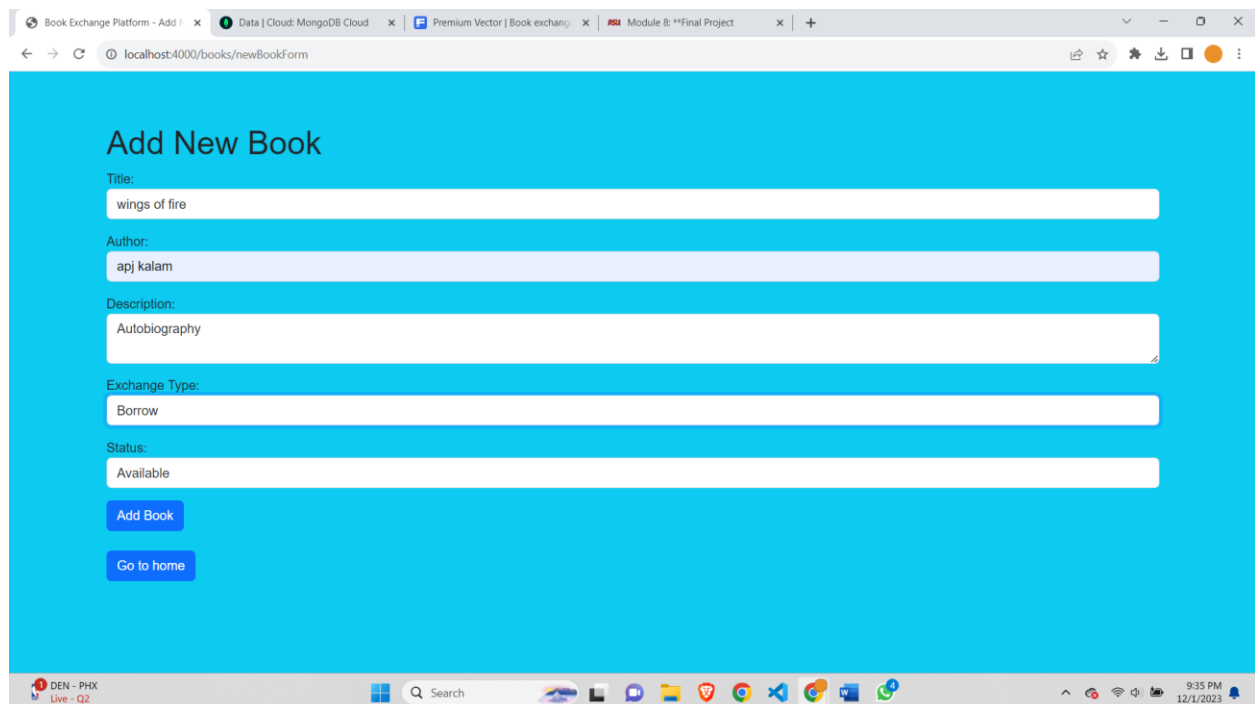
Signup Page-



After sign Up First Page (My Books List)-



Add book -



There are two options on add new Book incase the user just wishes to go back can click on go to home button.

After Adding Book-

The screenshot shows a web browser at localhost:4000/books/getBooks. The page has a teal header with the title 'My Book List' and buttons for 'Add New Book', 'Home', and 'Logout'. Below the header is a table with the following data:

Title	Author	Description	Exchange Type	Status	Actions
wings of fire	apj kalam	Autobiography	borrow	available	Edit Delete

In My books page we can have the option of editing and deleting the book.

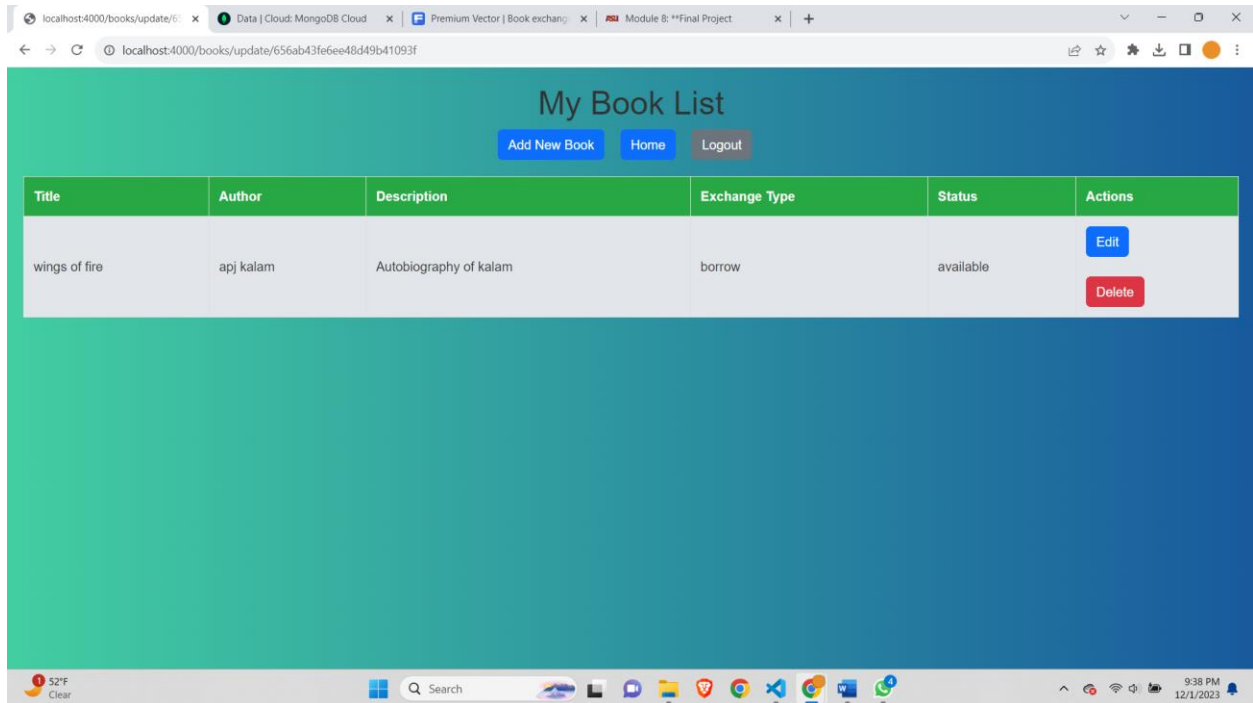
Edit book-

The screenshot shows the 'wings of fire' book edit page. The title 'wings of fire' is displayed at the top. Below it is the heading 'Update Book Exchange Entry'. The form contains the following fields:

- Title: wings of fire
- Author: apj kalam
- Description: Autobiography of kalam
- Exchange Type: Borrow
- Status: Available

At the bottom of the form are two buttons: 'Update' and 'Delete'.

After editing the book-

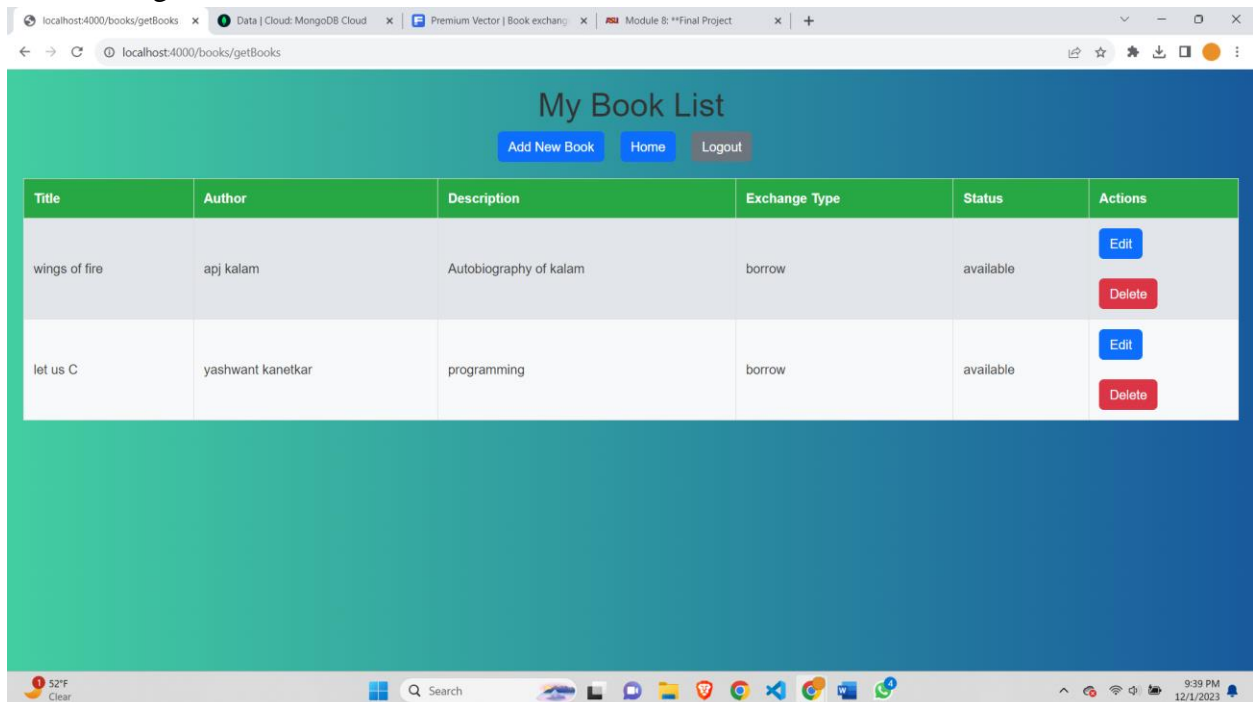


The screenshot shows a web browser window with the URL `localhost:4000/books/update/656ab43fe6ee48d49b41093f`. The page title is "My Book List". Below the title are three buttons: "Add New Book", "Home", and "Logout". A table displays a single book entry:

Title	Author	Description	Exchange Type	Status	Actions
wings of fire	apj kalam	Autobiography of kalam	borrow	available	Edit Delete

The Windows taskbar at the bottom shows the date and time as 9:38 PM on 12/1/2023.

After adding one more book-

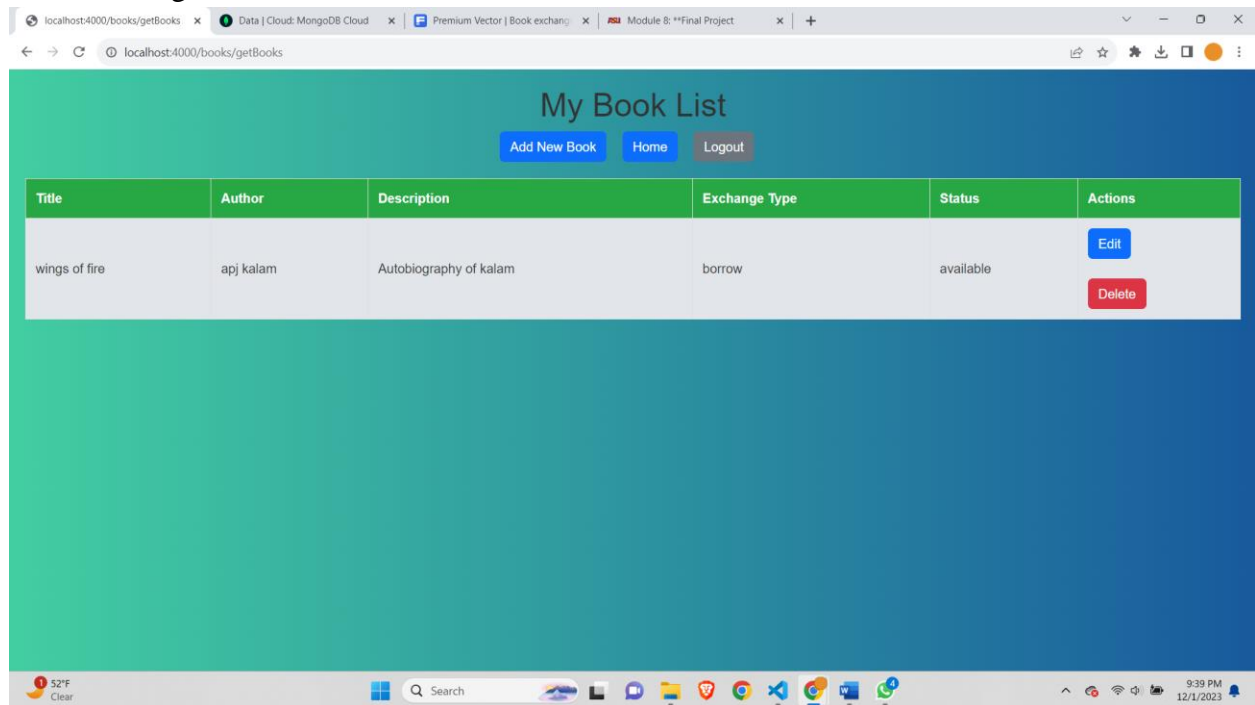


The screenshot shows the same web browser window, but the URL is `localhost:4000/books/getBooks`. The page title is "My Book List". Below the title are three buttons: "Add New Book", "Home", and "Logout". A table displays two book entries:

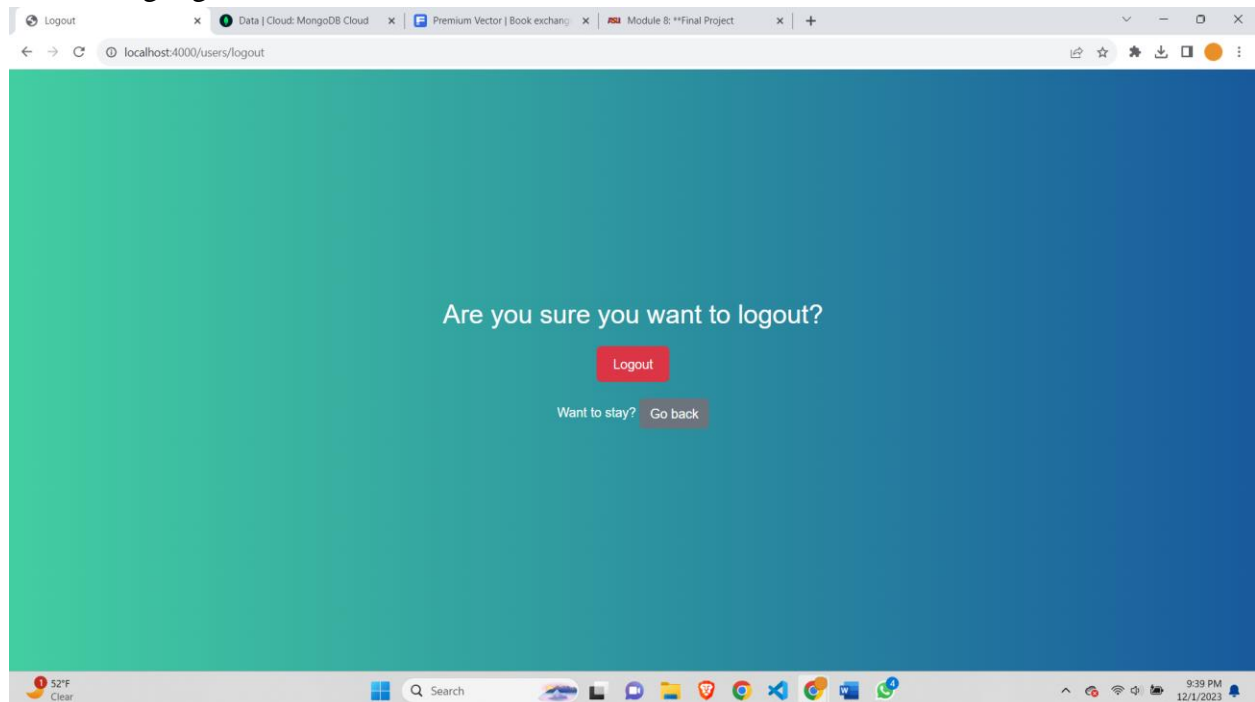
Title	Author	Description	Exchange Type	Status	Actions
wings of fire	apj kalam	Autobiography of kalam	borrow	available	Edit Delete
let us C	yashwant kanetkar	programming	borrow	available	Edit Delete

The Windows taskbar at the bottom shows the date and time as 9:39 PM on 12/1/2023.

After deleting one book-



On clicking logout-



Second User registration-

Register

Name:
test2

Email:
test2@gmail.com

Password:

Confirm Password:

Signup

Already have an account? [login](#)

Add Book-

Add New Book

Title:
dairy of anne frank

Author:
anne frank

Description:
autobiography

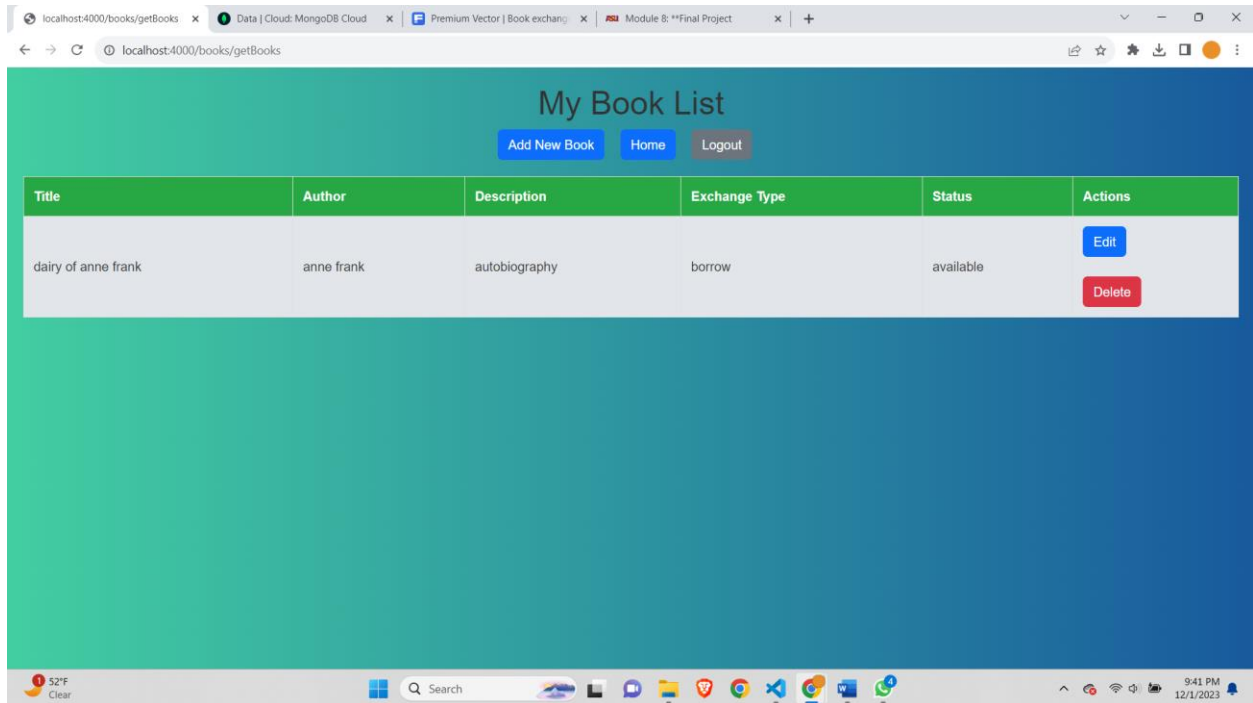
Exchange Type:
Borrow

Status:
Available

Add Book

Go to home

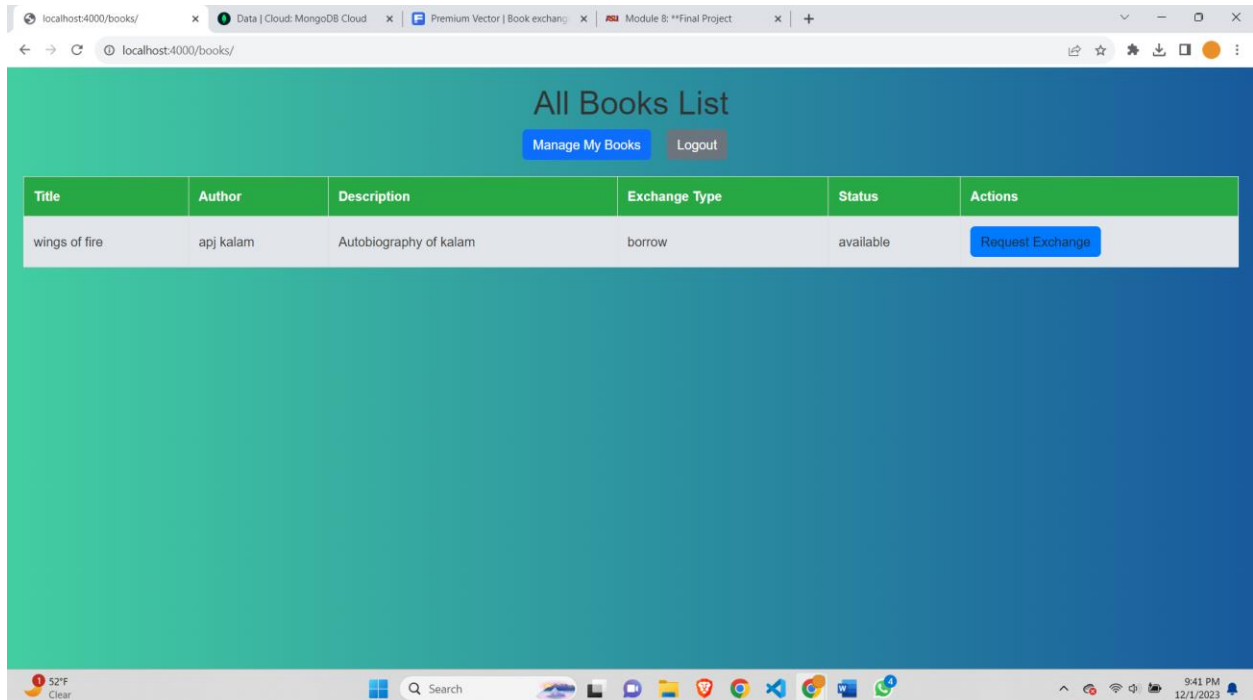
After adding book in test2-



The screenshot shows a web browser window with the URL `localhost:4000/books/getBooks`. The page title is "My Book List". Below the title are three buttons: "Add New Book" (blue), "Home" (blue), and "Logout" (grey). The main content is a table with the following columns: Title, Author, Description, Exchange Type, Status, and Actions. The table contains one row with the following data: Title: "dairy of anne frank", Author: "anne frank", Description: "autobiography", Exchange Type: "borrow", Status: "available", and Actions: "Edit" (blue button) and "Delete" (red button). The background of the page is a gradient of teal and blue. The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray shows the temperature as 52°F, the date as 12/1/2023, and the time as 9:41 PM.

Title	Author	Description	Exchange Type	Status	Actions
dairy of anne frank	anne frank	autobiography	borrow	available	<button>Edit</button> <button>Delete</button>

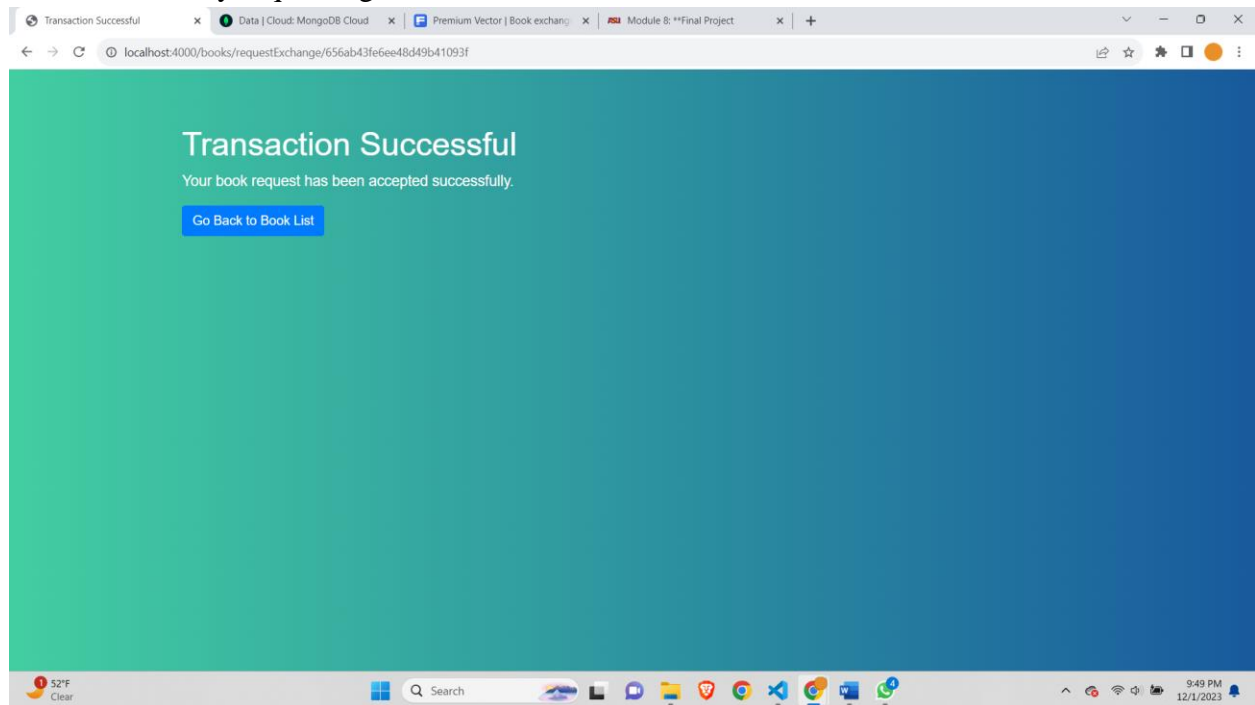
Requesting for book of different user (test1) we are test2 in this case-



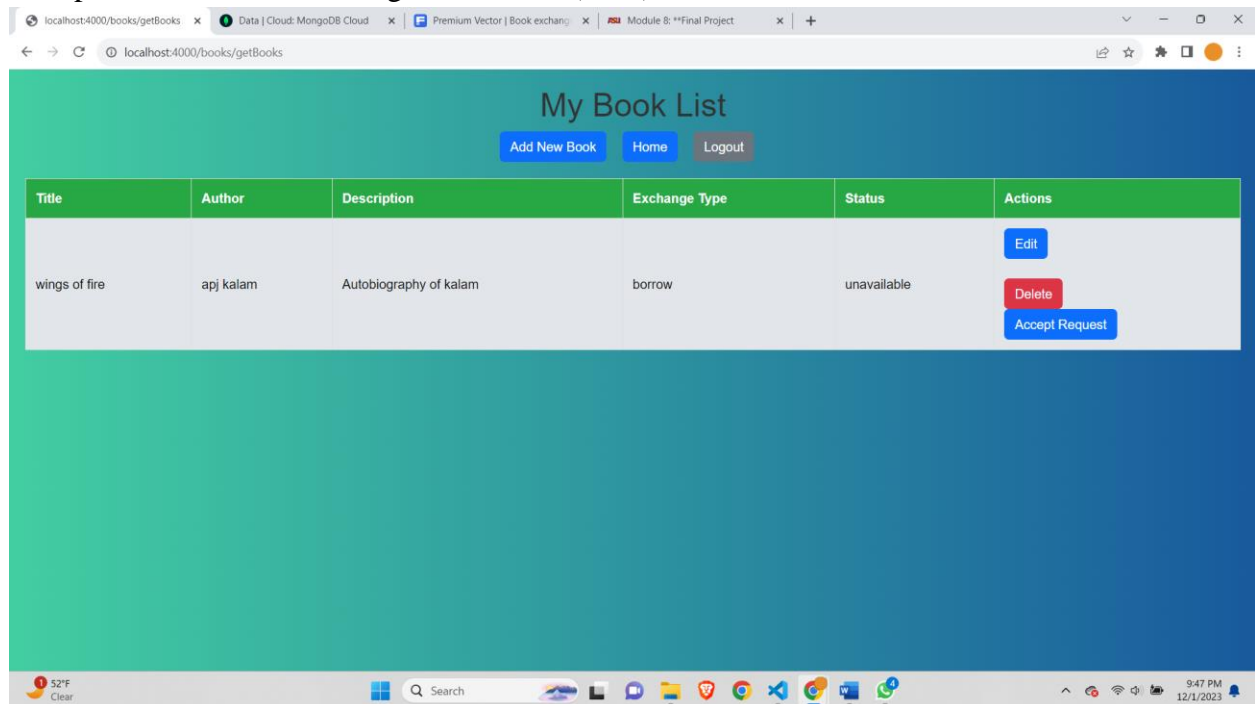
The screenshot shows a web browser window with the URL `localhost:4000/books/`. The page title is "All Books List". Below the title are two buttons: "Manage My Books" (blue) and "Logout" (grey). The main content is a table with the following columns: Title, Author, Description, Exchange Type, Status, and Actions. The table contains one row with the following data: Title: "wings of fire", Author: "apj kalam", Description: "Autobiography of kalam", Exchange Type: "borrow", Status: "available", and Actions: "Request Exchange" (blue button). The background of the page is a gradient of teal and blue. The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray shows the temperature as 52°F, the date as 12/1/2023, and the time as 9:41 PM.

Title	Author	Description	Exchange Type	Status	Actions
wings of fire	apj kalam	Autobiography of kalam	borrow	available	<button>Request Exchange</button>

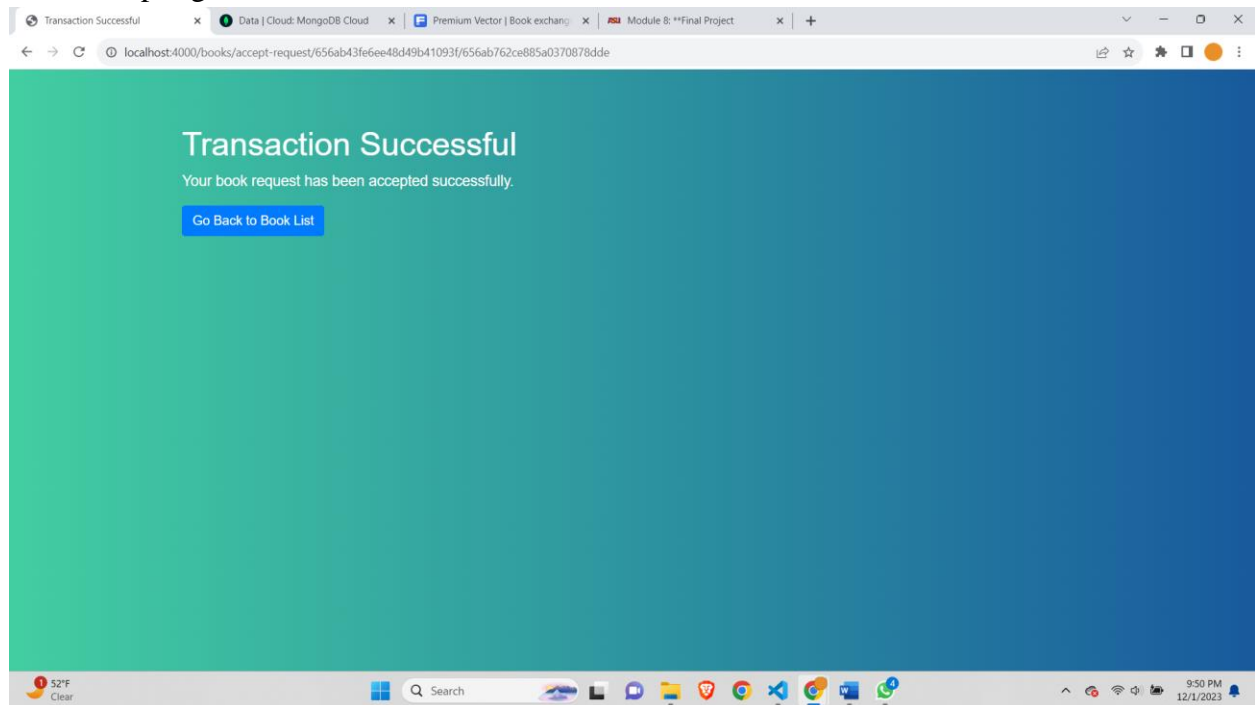
After Successfully requesting-



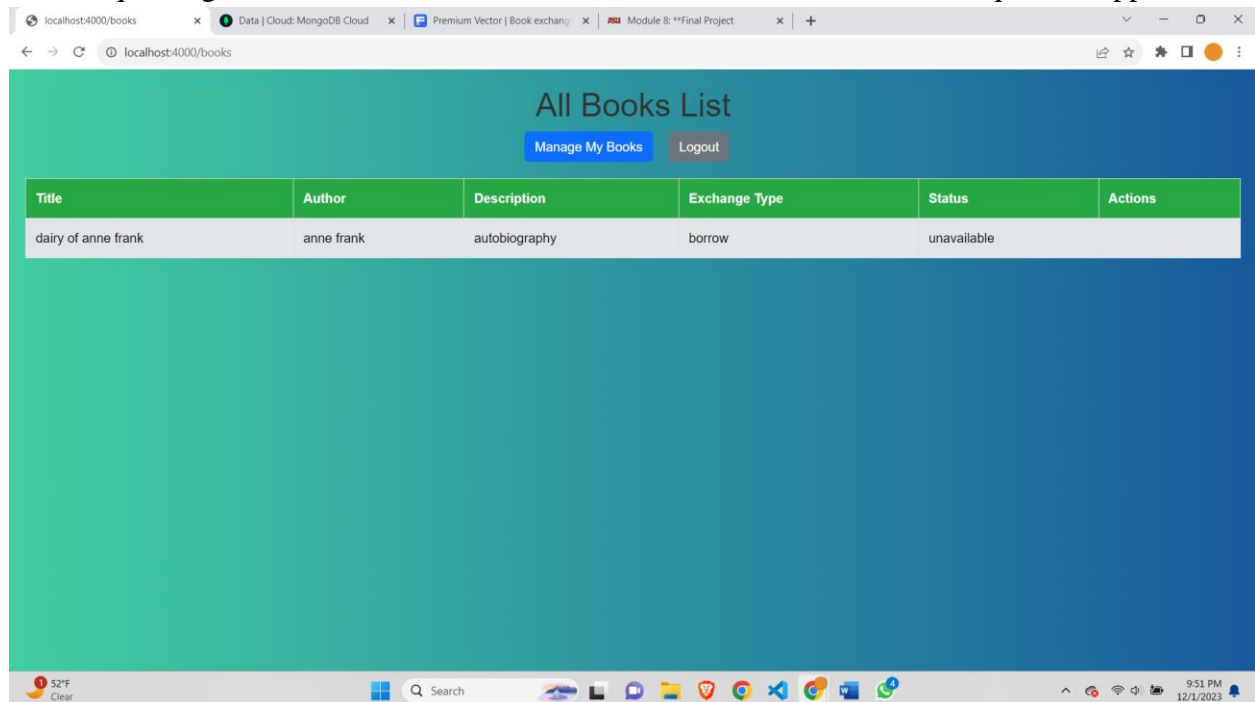
Accept the transaction from again first user (test1)-



After accepting the test cases-



After requesting for book the book becomes unavailable and the button to request disappears-



About page-


Browser tabs: About - Book Exchange Platform, Data | Cloud: MongoDB Cloud, Premium Vector | Book exchang, Module 8: **Final Project


Address bar: localhost:4000/users/about

About Book Exchange

[Home](#)

Student Information

Student Image: 

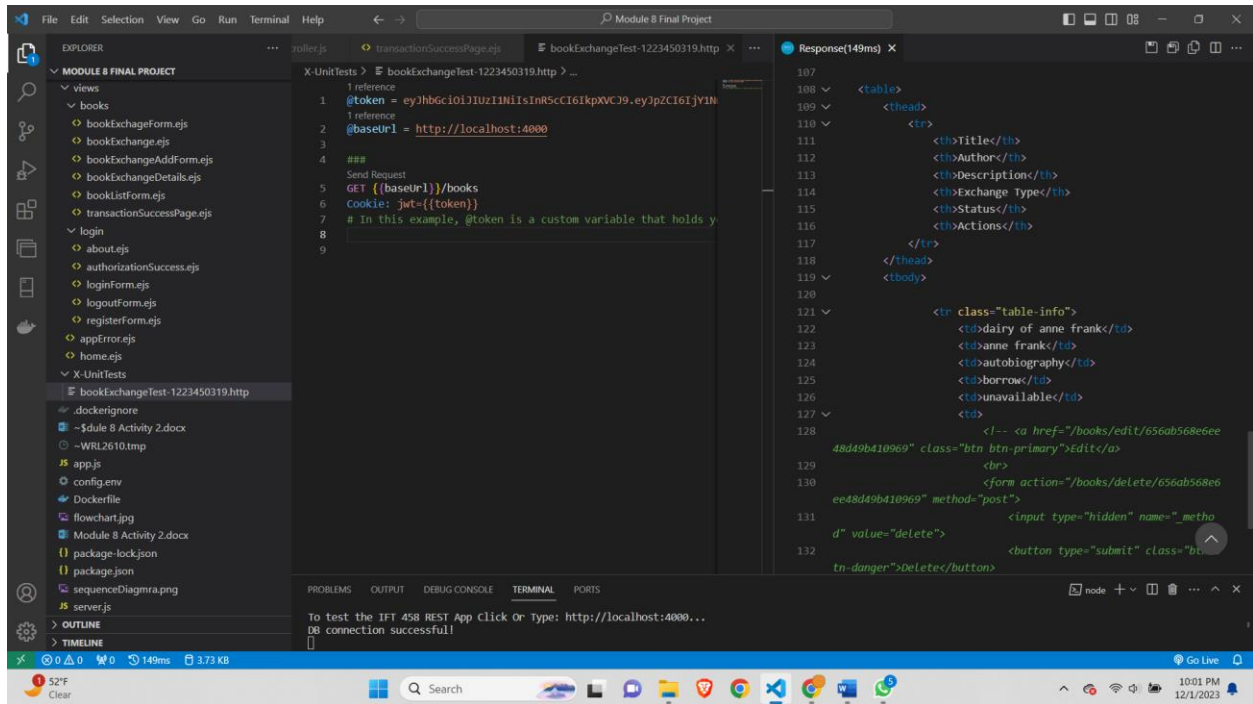
Student Name: Saurabh Kale
Student Email: skale12@asu.edu
Student ID: 1223450319
Course #: IFT 458/554
Batch of IFT 554 Fall 2023 

Server:
12/1/2023, 9-53:03 PM
Your IP address is: 75.204.18.33

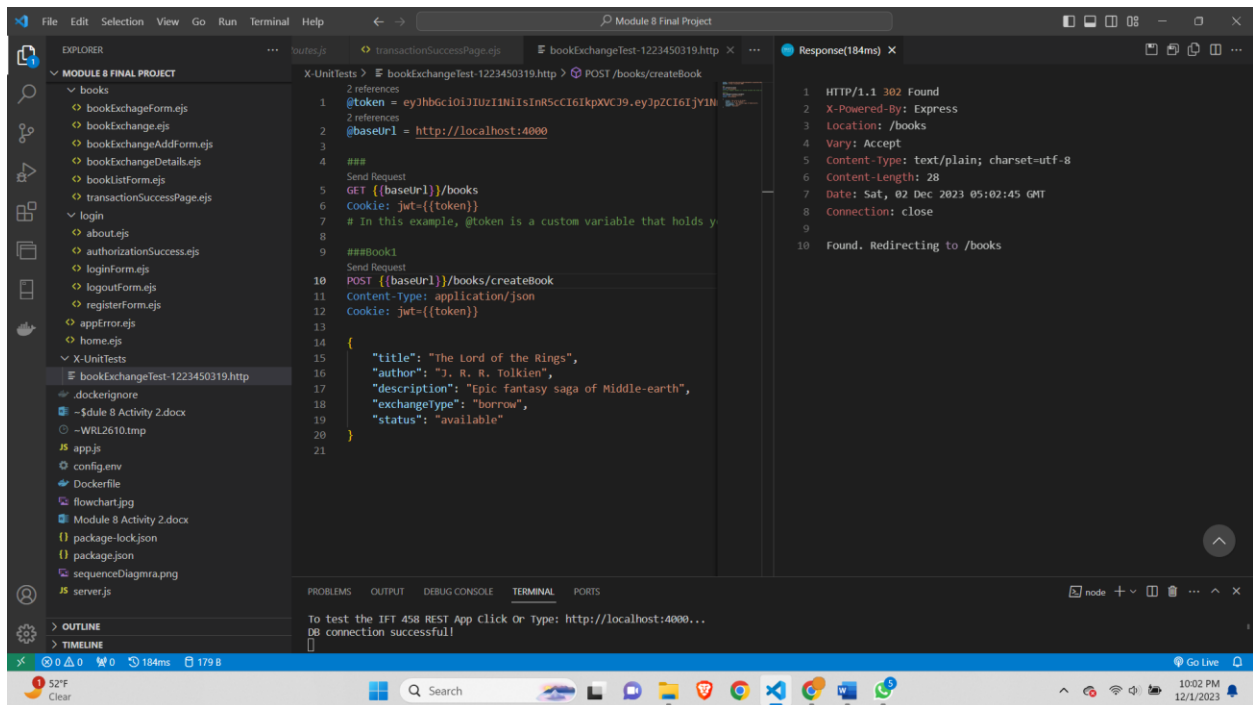
Book Exchange Platform © IFT 554 - Fall 2023

Taskbar: 52°F Clear, Search, File Explorer, Microsoft Edge, Google Chrome, Microsoft Word, Microsoft Teams, 9:53 PM 12/1/2023

Get Books will get books for this user only-



Create Books-



Book Added-

The screenshot shows the MongoDB Atlas web interface. The left sidebar contains navigation options: Overview, DEPLOYMENT, Database, Data Lake, SERVICES, Device Sync, Triggers, Data API, Data Federation, Search, Stream Processing, SECURITY, Backup, Database Access, Network Access, and Advanced. The 'Database' section is expanded, showing a list of databases including 1223450319_IFT_554. The main panel displays the details of the 1223450319_IFT_554 database, including storage size (32KB), logical data size (0B), total documents (0), and index total size (36KB). A document is shown with the following fields: `updatedAt` (2023-12-02T04:51:39.491+00:00), `_id` (1), `title` ("The Lord of the Rings"), `author` ("J. R. R. Tolkien"), `description` ("Epic fantasy saga of Middle-earth"), `exchangeType` ("borrow"), `owner` (ObjectId("656ab3a86ee48d49b410937")), `status` ("available"), `createdBy` (ObjectId("656ab3a86ee48d49b410937")), `requests` (Array (empty)), `createdAt` (2023-12-02T05:02:45.294+00:00), and `updatedAt` (2023-12-02T05:02:45.294+00:00).

Get specific Book-

The screenshot shows a VS Code editor with a REST client request and response. The request is a GET request to `http://localhost:4000/books/656ab3fe6ee48d49b41093f` with a `Cookie: jwt={{token}}` header. The response is a 200 OK status with a JSON body containing the book details: `{ "title": "The Lord of the Rings", "author": "J. R. R. Tolkien", "description": "Epic fantasy saga of Middle-earth", "exchangeType": "borrow", "status": "available" }`. The response is also shown in HTML format, displaying the book title and author.

Update The Book-

The screenshot shows the VS Code interface with the REST client open. The request is a POST to `http://localhost:4000/books/update/656ab43fe6ee48d49b41093f` with a JSON body. The response is a 267ms HTML document showing the updated book details.

```
18 "exchangeType": "borrow",
19 "status": "available"
20 }
21
22 ### Sample Test: Fetch book by ID
23 ##Book1
24 Send Request
25 GET http://localhost:4000/books/656ab43fe6ee48d49b41093f HTTP/1.1
26 Cookie: jwt={{token}}
27
28 ### Sample Test: Modify a book by ID
29 ##Book1
30 Send Request
31 POST http://localhost:4000/books/update/656ab43fe6ee48d49b41093f HTTP/1.1
32 Content-Type: application/json
33 Cookie: jwt={{token}}
34 {
35   "title": "Midnight Serenade",
36   "author": "Olivia Blackwell",
37   "description": "Music, passion, and moonlit romance.",
38   "exchangeType": "borrow",
39   "status": "available"
40 }
41
```

```
109 <table>
110 <thead>
111 <tr>
112 <th>Title</th>
113 <th>Author</th>
114 <th>Description</th>
115 <th>Exchange Type</th>
116 <th>Status</th>
117 <th>Actions</th>
118 </tr>
119 </thead>
120 <tbody>
121
122
123 <tr class="">
124 <td>Midnight Serenade</td>
125 <td>Olivia Blackwell</td>
126 <td>Music, passion, and moonlit romance.</td>
127 <td>borrow</td>
128 <td>available</td>
129 <td>
130 <a href="/books/edit/656ab43fe6ee48d49b41093f" class="btn btn-primary">Edit</a>
131 <br>
132 <form action="/books/delete/656ab43fe6ee48d49b41093f" method="post">
133 <input type="hidden" name="_method" value="delete">
134 <button type="submit" class="btn btn-danger">Delete</button>
135 </form>
136 </td>
137 </tr>
138 </tbody>
139 </table>
```

Update using PUT-

The screenshot shows the VS Code interface with the REST client open. The request is a PUT to `http://localhost:4000/books/update/656ab43fe6ee48d49b41093f` with a JSON body. The response is a 32ms HTML document showing the updated book details.

```
29 ## Sample Test: Modify a book by ID
30 ##Book1
31 Send Request
32 PUT http://localhost:4000/books/update/656ab43fe6ee48d49b41093f HTTP/1.1
33 Content-Type: application/json
34 Cookie: jwt={{token}}
35 {
36   "title": "Midnight Serenade",
37   "author": "Olivia Blackwell",
38   "description": "Music, passion, and moonlit romance.",
39   "exchangeType": "borrow",
40   "status": "available"
41 }
42
43 ### Sample Test: Modify a book by ID
44 ##Book1
45 PUT http://localhost:4000/books/update/656ab43fe6ee48d49b41093f HTTP/1.1
46 Content-Type: application/json
47 Cookie: jwt={{token}}
48 {
49   "title": "Midnight Serenade 22",
50   "author": "Olivia Blackwell",
51   "description": "Music, passion, and moonlit romance.",
52   "exchangeType": "borrow",
53   "status": "available"
54 }
55
56 ### Sample Test: Delete book by ID
57 ##Book1
58 Send Request
59 DELETE http://localhost:4000/books/656ab43fe6ee48d49b41093f HTTP/1.1
60 Cookie: jwt={{token}}
```

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 2332
5 ETag: W/"91c-4kiR2IVC48BAjrhDkS0wPqitg"
6 Date: Sat, 02 Dec 2023 05:08:53 GMT
7 Connection: close
8
9 <!DOCTYPE html>
10 <html lang="en">
11
12 <head>
13 <title>Login</title>
14 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2VBKqghggwzi7pPCaAQ46GNC80w1R" crossorigin="anonymous">
15 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBIe4zVF0s0GIM5b4tccpxy9F7jl+j" crossorigin="anonymous"></script>
16
17 <style>
18   body {
19     font-family: Arial, sans-serif;
20     padding: 20px;
21     margin: 0;
22     height: 100vh;
23     display: flex;
24
```


Delete using delete method-

The screenshot shows a VS Code editor with a REST client project named 'Module 8 Final Project'. The Explorer panel on the left shows the project structure, including files like 'bookExchangeForm.ejs', 'bookExchange.ejs', 'bookExchangeAddForm.ejs', 'bookExchangeDetails.ejs', 'bookListForm.ejs', 'transactionSuccessPage.ejs', 'login', 'about.ejs', 'authorizationSuccess.ejs', 'loginForm.ejs', 'logoutForm.ejs', 'registerForm.ejs', 'appError.ejs', 'home.ejs', 'X-UnitTests', and 'bookExchangeTest-1223450319.http'. The main editor area shows the 'bookExchangeTest-1223450319.http' file with a REST client request and response.

Request:

```
39 "exchangeType": "borrow",
40 "status": "available"
41 }
42
43 ### Sample Test: Modify a book by ID
44 ###Book 1
45 Send Request
46 PUT http://localhost:4000/books/update/656ab43fe6ee48d49b41
47 Content-Type: application/json
48 Cookie: jwt={{token}}
49 {
50   "title": "Midnight Serenade 22",
51   "author": "Olivia Blackwell",
52   "description": "Music, passion, and moonlit romance.",
53   "exchangeType": "borrow",
54   "status": "available"
55 }
56
57 ### Sample Test: Delete book by ID
58 ### Book 1
59 Send Request
60 DELETE http://localhost:4000/books/delete/656ab43fe6ee48d49
61 Cookie: jwt={{token}}
```

Response (6ms):

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 2332
5 ETag: W/"91c-4kiR21VC48BAJkrHdkSU0wPqTtg"
6 Date: Sat, 02 Dec 2023 05:09:22 GMT
7 Connection: close
8
9 <!DOCTYPE html>
10 <html lang="en">
11
12 <head>
13   <title>Login</title>
14   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet"
15     integrity="sha384-rbsA2VBKqhgywzxi7pPcAqQ46MgncM80W1R
16     WUHG1DGLW2JE8K2KadqZf9CLG65" crossorigin="anonymous">
17   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/d
18     ist/js/bootstrap.bundle.min.js"
19     integrity="sha384-kenU1KFdBIe4zVF0s0G1MSb4hpcxy99F7JL+j
20     jXkk+Q2h455rYXK/7HAuoJl+0I4"
21     crossorigin="anonymous"></script>
22   <style>
23     body {
24       font-family: Arial, sans-serif;
25       padding: 20px;
26       margin: 0;
27       height: 100vh;
28       display: flex;
```

The bottom status bar shows the current state: 0 errors, 0 warnings, 0 info, 6ms execution time, and 2.44 KB size. The system tray at the bottom shows the temperature (52°F), search bar, and system clock (10:09 PM, 12/1/2023).

Implementation

Challenges faced-

- At some points I was not able to see the UI changes. For which the workaround was to give hard refresh which is ctrl + shift + r.
- Some typo mistakes were made and which had to be debugged using the console.log and sometimes the debugger terminal.
- Sitting at one point and trying to debug some errors for longer time didn't help few times. After taking a break and coming back to the code helped me fix it within mins as I was fresh and looking from different angle to the code.
- Doing the accept transaction and request transaction was one of the difficult part as it took some time thinking where and what we have to do.
- Comprehensive testing of various scenarios, including user authentication, book transactions, and error cases.

Key Functionalities-

1. User Authentication:

- Allowing users to sign up with a name, email, and password.
- Encrypting and securely storing user passwords using bcrypt.
- Enforcing password length requirements.
- Implementing a login mechanism that checks user credentials against stored data.
- Generating and sending JSON Web Tokens (JWT) upon successful authentication.

2. User Authorization:

- Assigning roles to users (client, staff, student, admin).
- Protecting routes by implementing middleware for authentication and authorization checks.
- Granting access based on user roles.

3. Book Management:

- Allowing authenticated users to add new books to the system.
- Defining a data model for books with attributes such as title, author, description, and exchange type.
- Setting default values for book status and ownership.
- Handling CRUD operations for books (Create, Read, Update, Delete).
- Displaying a list of books owned by the current user and books available for exchange.

4. Book Requests:

- Enabling users to request books from other users.
- Managing a requests array within each book, including user information and request status.
- Preventing users from requesting the same book multiple times.
- Changing the status of a book to 'pending' when a request is made.

5. Book Transactions:

- Allowing users to accept book requests, updating the request status to 'accepted.'
- Changing the owner of the book to the user who accepted the request.
- Displaying a transaction success page after a successful book request or acceptance.

6. User Interface (UI):

- Rendering views using the EJS templating engine.
- Providing a user-friendly interface for actions such as adding books, viewing book details, and managing requests.
- Displaying relevant information based on user roles and actions.

7. Error Handling:

- Implementing error handling for various scenarios, such as incorrect login credentials, duplicate book requests, and database errors.
- Rendering error pages with meaningful messages to users.

8. Security Measures:

- Incorporating JWT for secure user authentication.
- Using bcrypt for password hashing to enhance security.
- Implementing middleware for protecting routes and checking user roles.

9. Logout:

- Providing users with the ability to log out of their accounts.
- Clearing the JWT cookie upon logout.

Learning Outcomes

- In module 5 we learned the JWT token authorization, MongoDB, MVC architecture and using the config.env file which I could use all of these things in my project.
- In Module 6 we learned JWT token in depth and got good practice of using it.
- In Module 7 we understood the flow diagrams and implemented it in the project. The sequence Diagram, flowcharts,etc are learned and implemented in the project.
- Learn how to interact with a MongoDB database using Mongoose, an ODM (Object Data Modeling) library.
- Understand schema creation, model definition, and CRUD operations in MongoDB.
- Gain proficiency in using Node.js for server-side JavaScript development.
- Understand the fundamentals of Express.js, a popular web application framework for Node.js.

Constructive Feedback

The course takes good practice of code and step by step implementation of the topic. Although the professor tries to get doubts from students and the graders are too helpful. If we had to give a suggestion then my suggestion would be to start a different online class for students those who want to learn the things. As few of the students are scared to ask doubt. In online class in mid week professor or the grader can explain step by step what is the assignment and what things are needed to be done. As by the mid week students might start the assignment and can have doubts to ask.

References

Image was taken from-

Book exchange landing page template or bookcrossing vector illustration banner education and knowledge concept diverse hands holding books swap literature event library day culture festival Premium Vector. (2022, November 8). Freepik. https://www.freepik.com/premium-vector/book-exchange-landing-page-template-bookcrossing-vector-illustration-banner-education-knowledge-concept-diverse-hands-holding-books-swap-literature-event-library-day-culture-festival_34126356.htm

MongoDB documentation. MongoDB Documentation. <https://www.mongodb.com/docs/>

Documentation / Node.js. <https://nodejs.org/en/docs>