

**Module 7 Lab 1: Develop Secure Middleware & Backend Database for a Book Exchange  
Application**

**Brandon Trinkle**

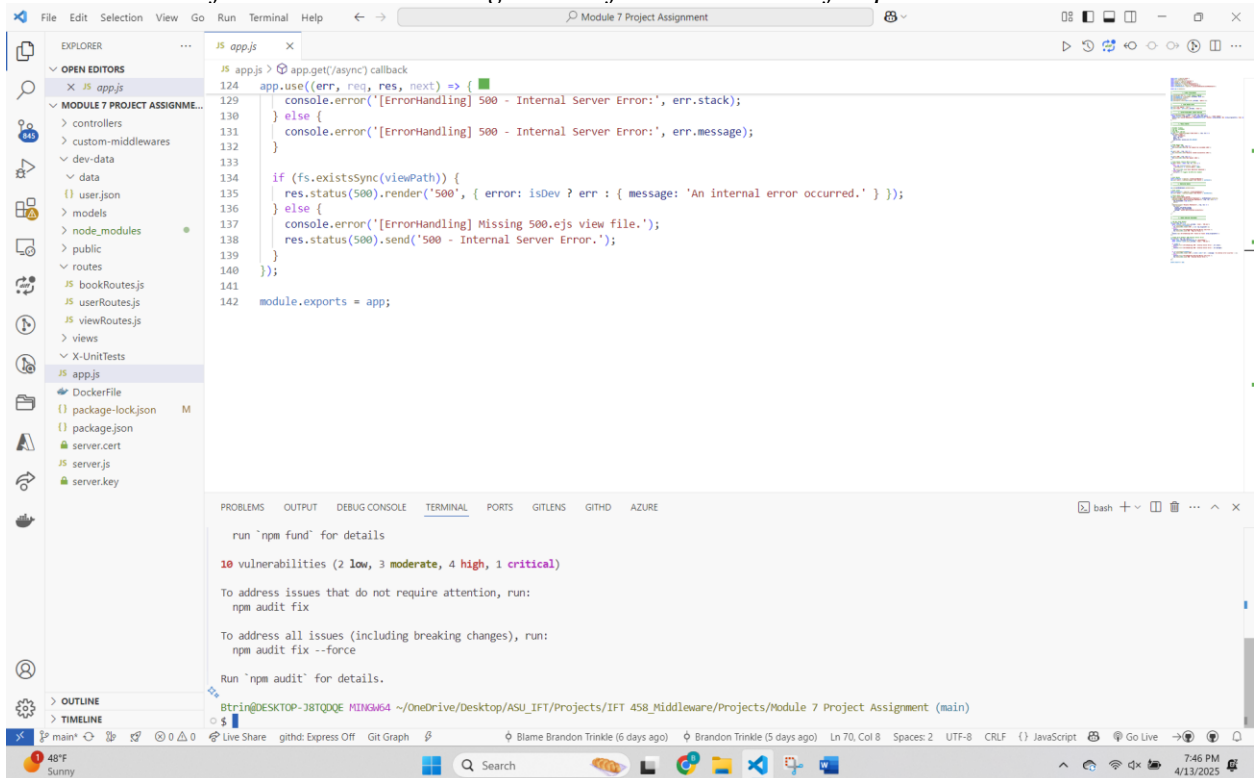
**IFT 458: Middleware Programming**

**Professor Dinesh Sthapit**

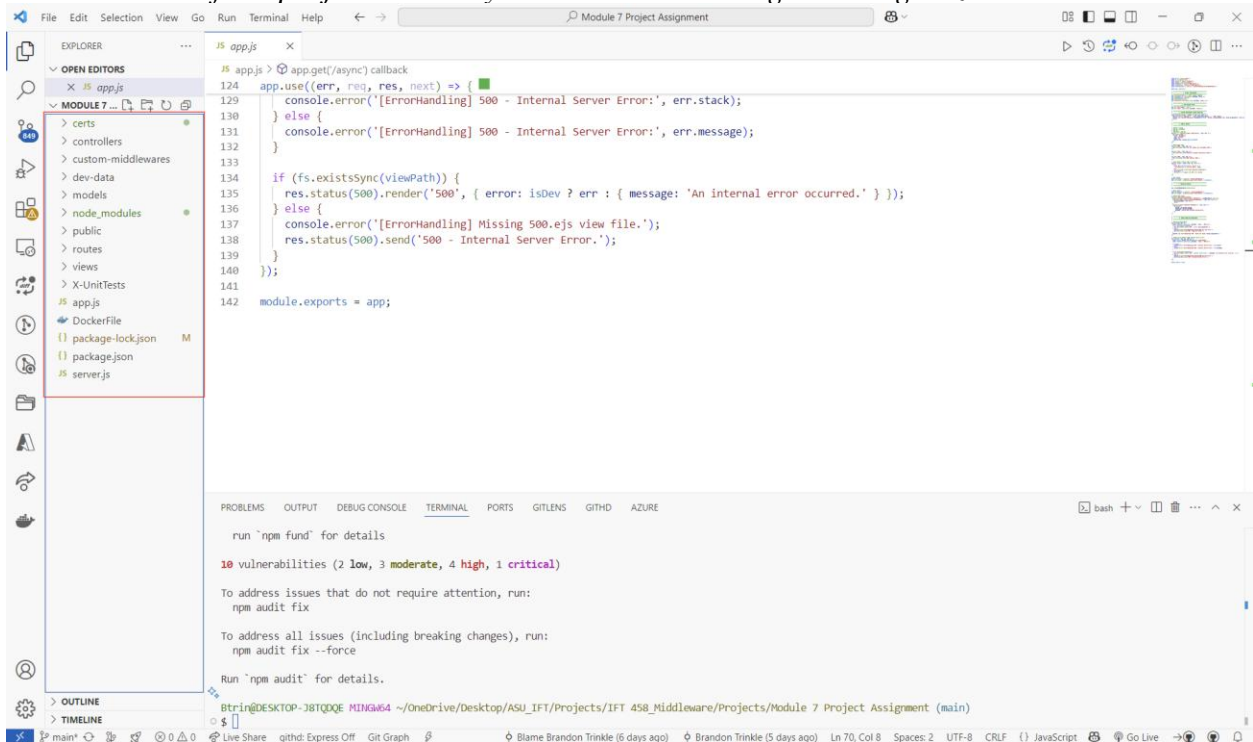
**April 13, 2025**

## Module 7 Lab 1: Develop Secure Middleware & Backend Database for a Book Exchange Application

### 1. Screenshot 1: Of the terminal showing successful installation of dependencies.

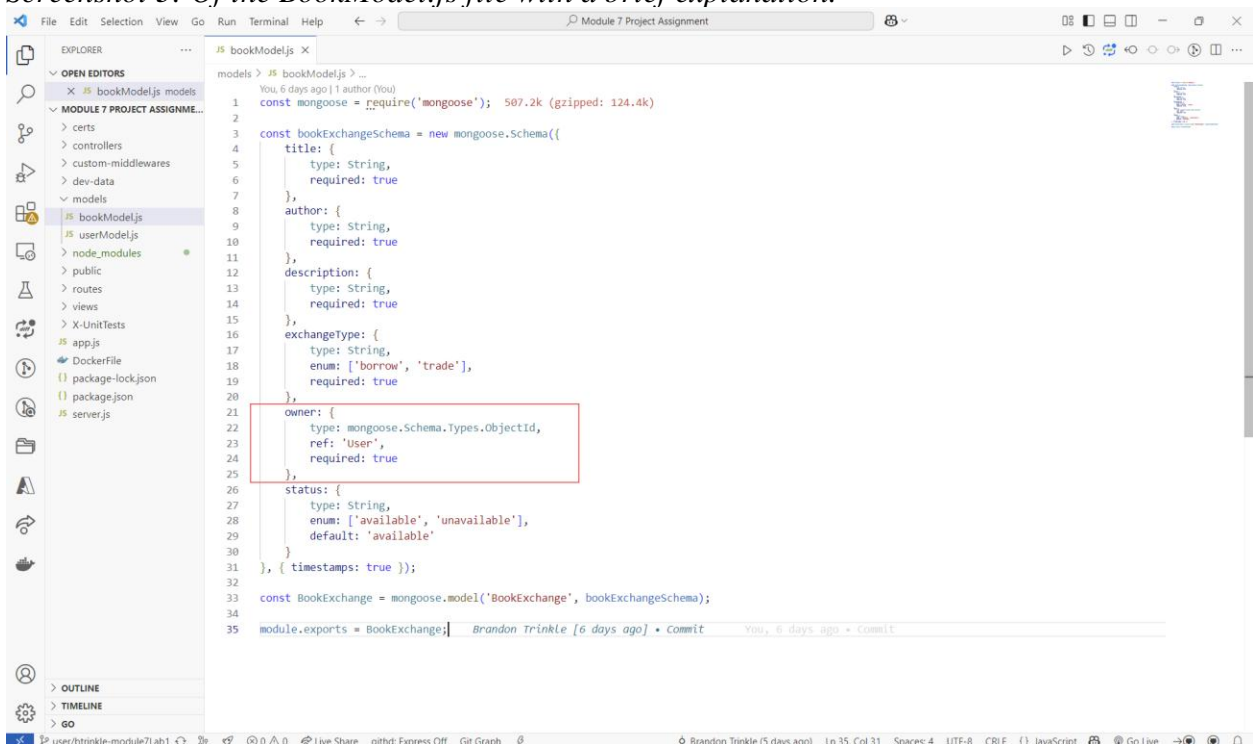


2. Screenshot 2: Of the project directory structure showcasing MVC organization.



*Note: This structure follows the MVC pattern. Certs created a few lessons ago are stored in the certs directory.*

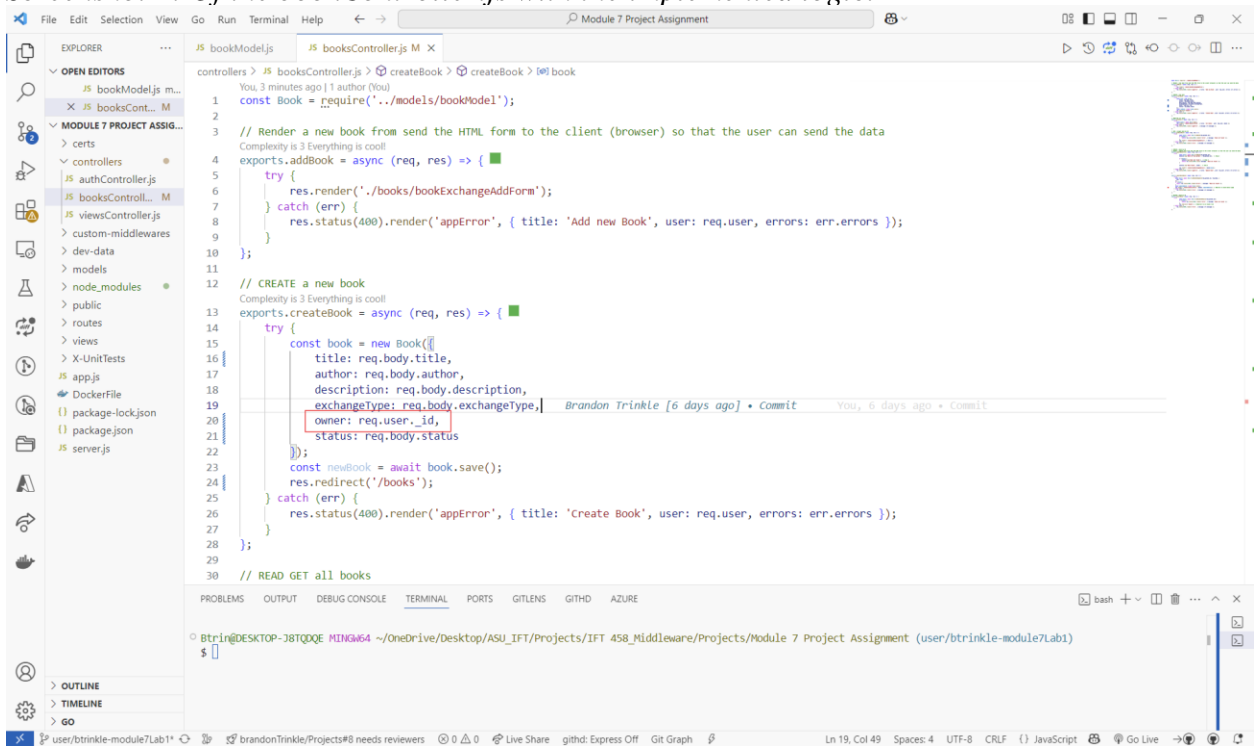
3. Screenshot 3: Of the BookModel.js file with a brief explanation.



*Note: BookModel.js file defines a Mongoose schema named bookExchangeSchema that*

establishes the structure for storing book exchange data in MongoDB. Each book document includes required fields such as title, author, and description, ensuring that all necessary details are provided. Regarding the question in this section; we already store userID of the user who added the book with 'owner'.

#### 4. Screenshot 4: Of the bookController.js with the implemented logic.

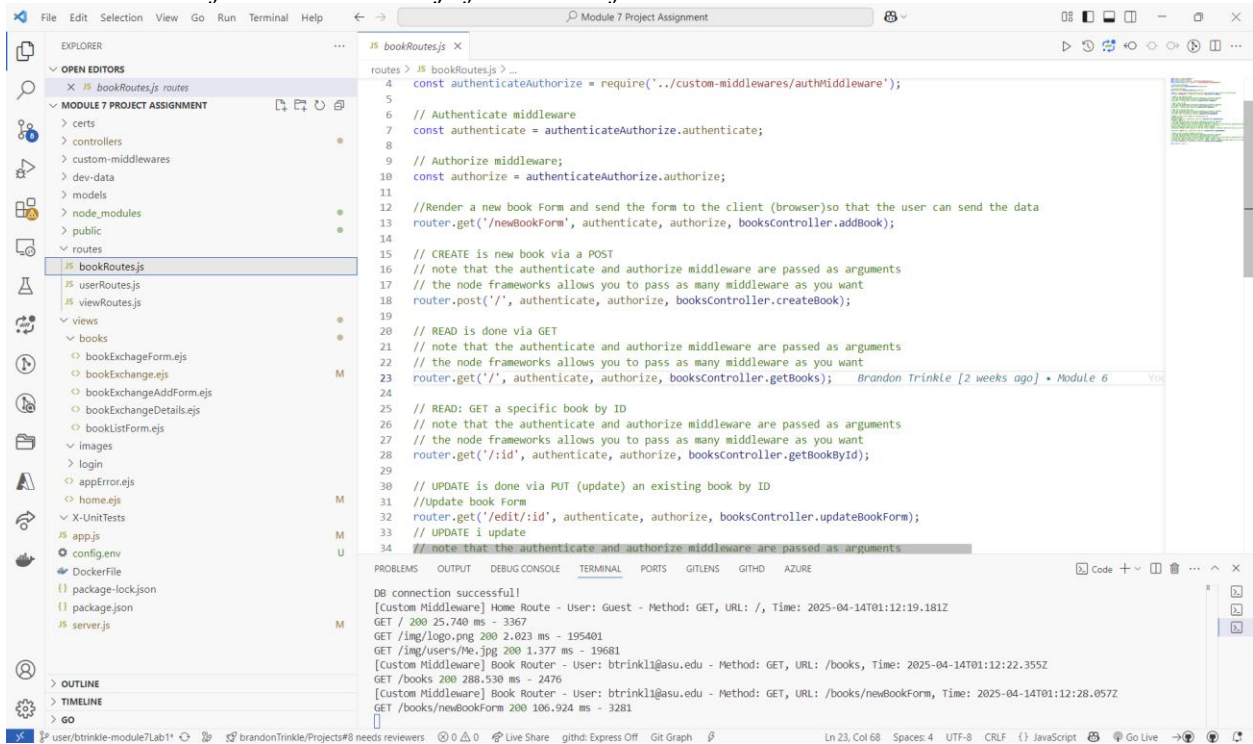


```

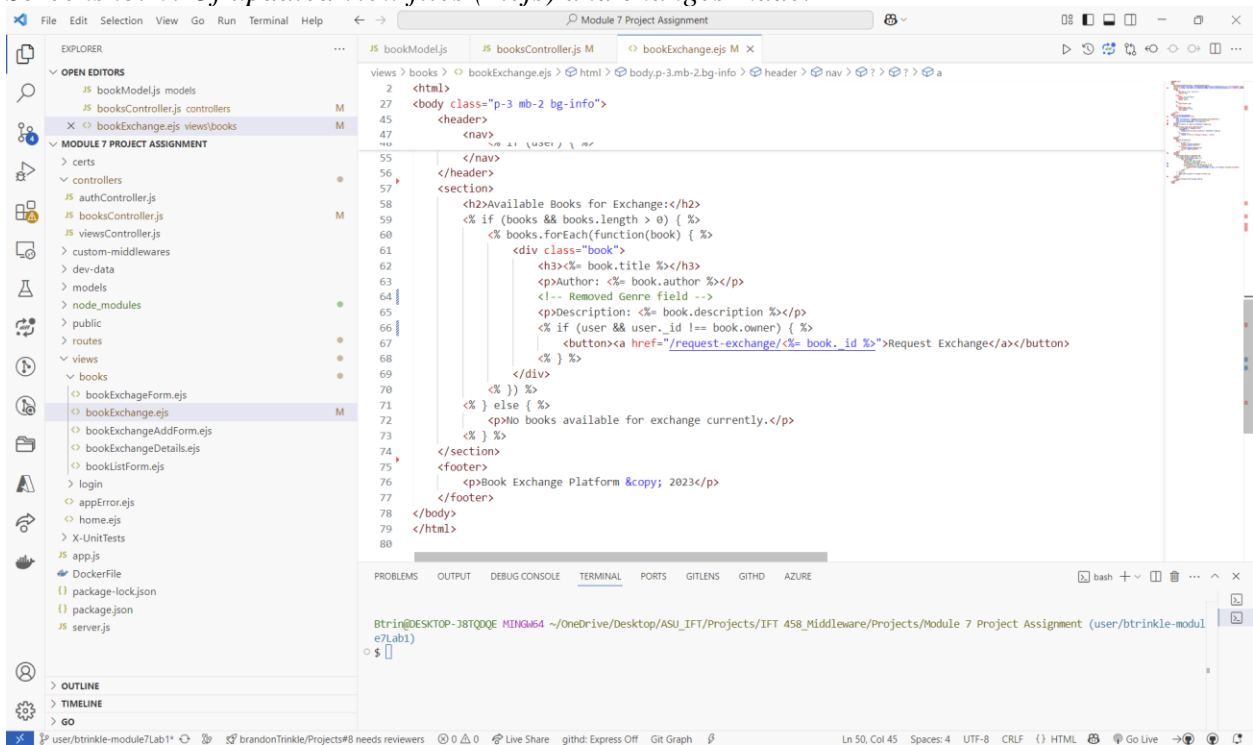
1  const Book = require('../models/bookModel');
2
3  // Render a new book form to send the HTML form to the client (browser) so that the user can send the data
4  exports.addBook = async (req, res) => {
5    try {
6      res.render('./books/bookExchangeAddForm');
7    } catch (err) {
8      res.status(400).render('appError', { title: 'Add new Book', user: req.user, errors: err.errors });
9    }
10  };
11
12  // CREATE a new book
13  exports.createBook = async (req, res) => {
14    try {
15      const book = new Book({
16        title: req.body.title,
17        author: req.body.author,
18        description: req.body.description,
19        exchangeType: req.body.exchangeType,
20        owner: req.user._id,
21        status: req.body.status
22      });
23      const newBook = await book.save();
24      res.redirect('/books');
25    } catch (err) {
26      res.status(400).render('appError', { title: 'Create Book', user: req.user, errors: err.errors });
27    }
28  };
29
30  // READ GET all books
  
```

Note: owner exists already when created a new book as shown in bookController.js

## 5. Screenshot 5: Of the bookRoutes.js file with defined routes.

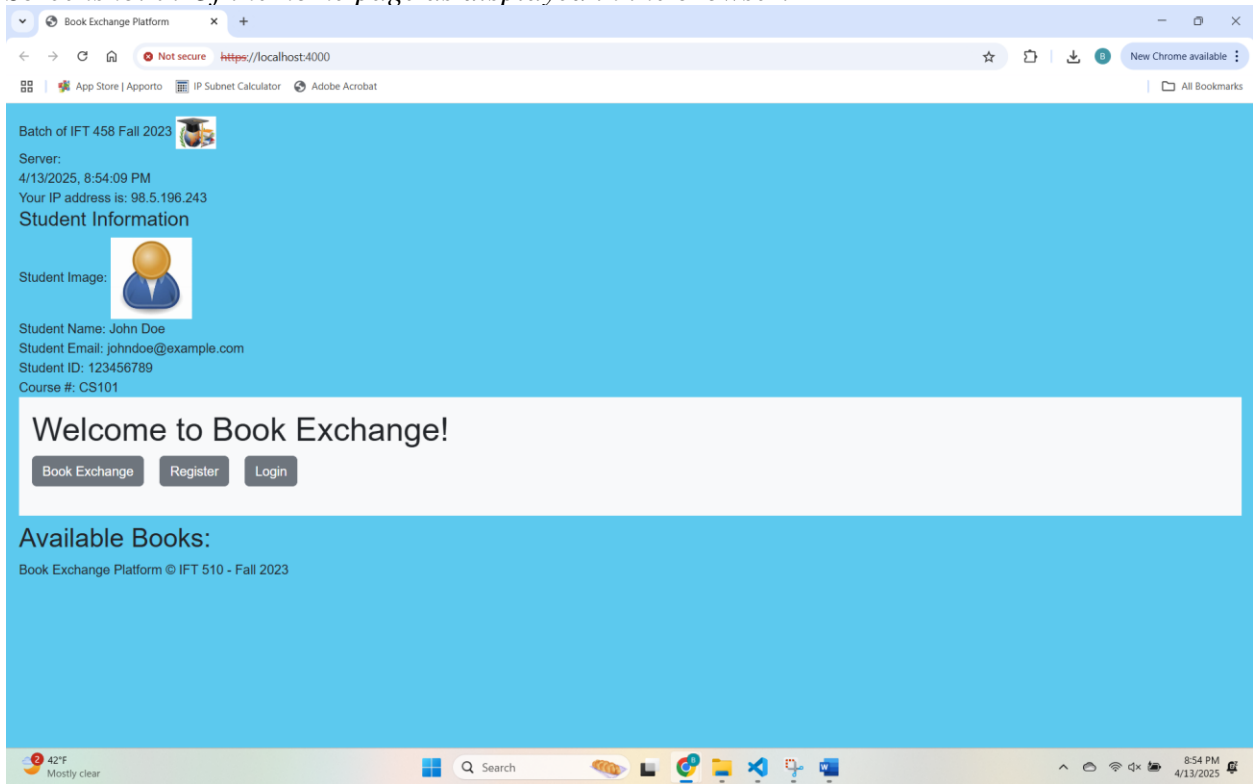


## 6. Screenshot 6: Of updated view files (\*.ejs) and changes made.

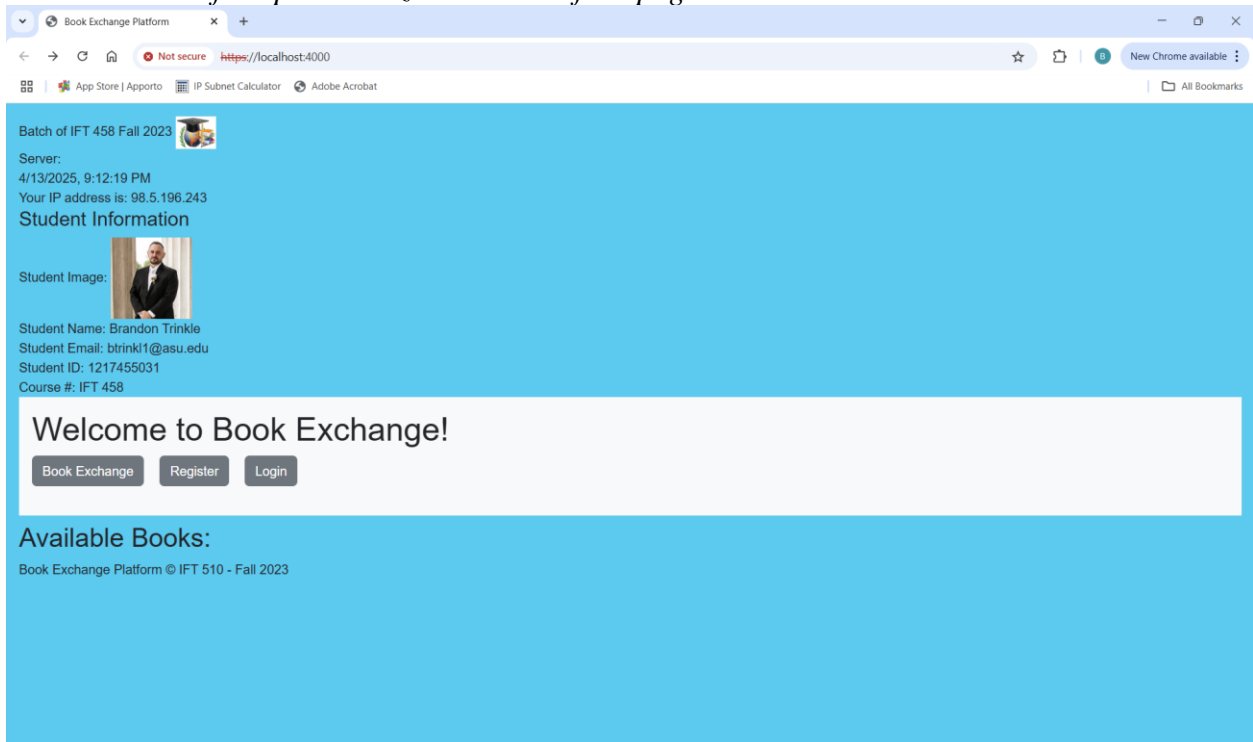


Note: I removed the Genre section as its not defined in our model.

7. *Screenshot 7: Of the home page as displayed in the browser.*

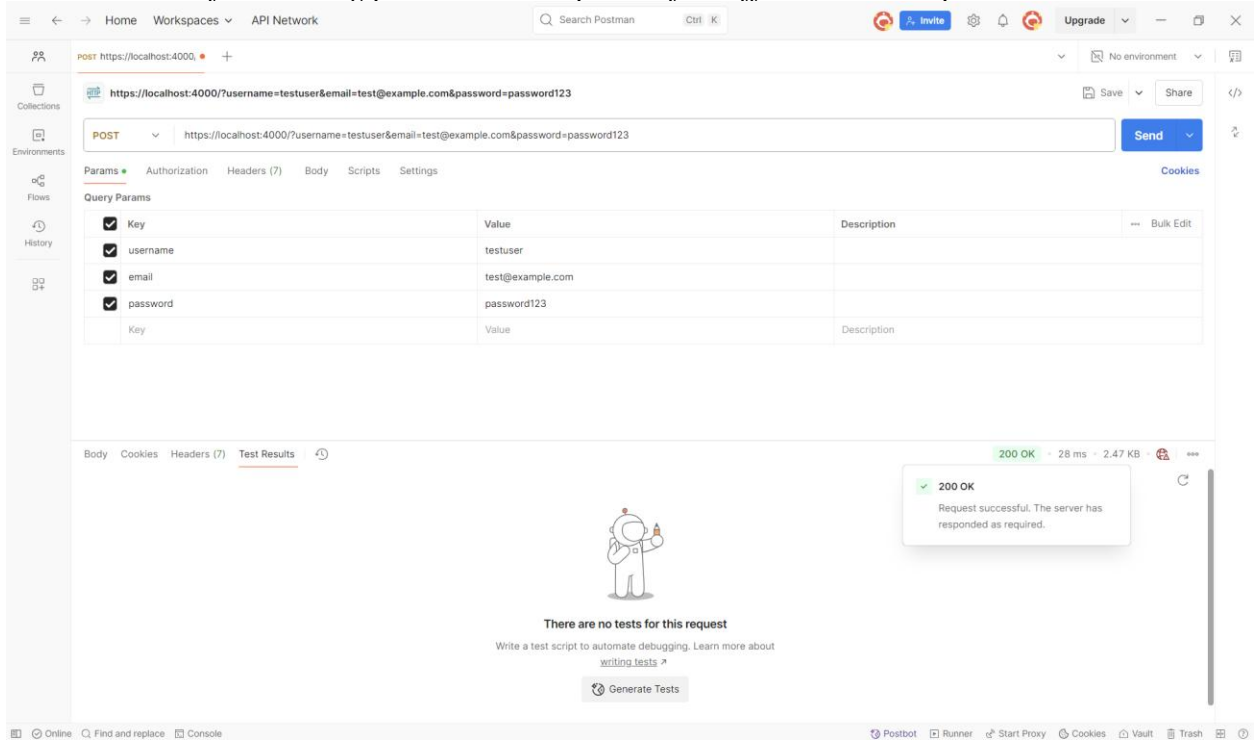


8. *Screenshot 8: Of all personalized user interface pages.*



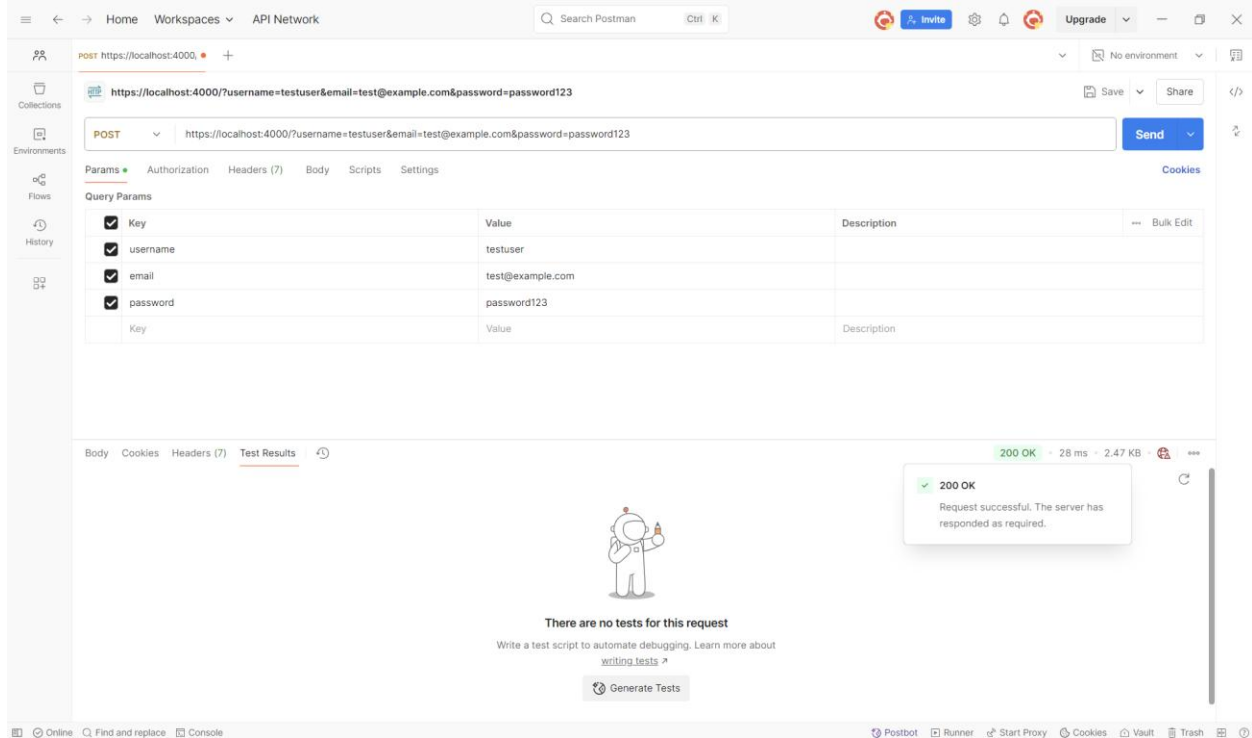
*Note: home.ejs is the only file that is customized to the user*

9. Screenshot 9: Of the testing process and responses for different API endpoints.



10. Screenshot 10: Of API specification documentation and screenshots of the corresponding code for each endpoint.

a. Register / Sign up



## b. Login

The screenshot shows the Postman interface for a POST request to `https://localhost:4000/?email=test@example.com&password=password123`. The request is successful, returning a `200 OK` status with a response time of 21 ms and a body size of 2.47 KB. The response body contains a cartoon character and the message "There are no tests for this request".

**Query Params**

Key	Value	Description
username	testuser	
email	test@example.com	
password	password123	

**Test Results**

200 OK · 21 ms · 2.47 KB

There are no tests for this request

Write a test script to automate debugging. Learn more about [writing tests](#)

[Generate Tests](#)

## c. Protected Resource

The screenshot shows the Postman interface for a GET request to `https://localhost:4000/api/protected`. The request is successful, returning a `200 OK` status with a response time of 25 ms and a body size of 2.47 KB. The response body contains a cartoon character and the message "There are no tests for this request".

**Headers**

Key	Value	Description
Bearer	IVFgPGisucpwqBkn	

**Test Results**

200 OK · 25 ms · 2.47 KB

There are no tests for this request

Write a test script to automate debugging. Learn more about [writing tests](#)

[Generate Tests](#)



### d. View All Books

The screenshot shows the Postman interface for a GET request to `https://localhost:4000/api/books`. The request is configured with the following test scripts:

```

1 pm.test("Response status code is 200", function () {
2   pm.expect(pm.response.code).to.equal(200);
3 });
4
5
6 pm.test("Response time is less than 200ms", function () {
7   pm.expect(pm.response.responseTime).to.be.below(200);
8 });
9
10
11 pm.test("Response must have the required Content-Type header", function () {
12   pm.expect(pm.response.headers.get("Content-Type")).to.include("text/html");
13 });
14
15
16 pm.test("Response body is not empty", function () {
17   pm.expect(pm.response.text()).to.not.be.empty;
18 });

```

The response is a **200 OK** status with a response time of 24 ms and a body size of 2.47 KB. The test results show all five tests passed:

- PASSED** Response status code is 200
- PASSED** Response time is less than 200ms
- PASSED** Response must have the required Content-Type header
- PASSED** Response body is not empty
- PASSED** Response body is in valid HTML format

### e. Add Book Exchange

The screenshot shows the Postman interface for a GET request to `https://localhost:4000/api/books/newBookForm`. The request is configured with the following query parameters:

Key	Value	Description
username	testuser	
password	password123	

The response is a **200 OK** status with a response time of 10 ms and a body size of 2.47 KB. The test results show all five tests passed:

- PASSED** Response status code is 200
- PASSED** Response time is less than 200ms
- PASSED** Response must have the required Content-Type header
- PASSED** Response body is not empty
- PASSED** Response body is in valid HTML format

## *f. Update Delete Exchange*

The screenshot shows the Postman API client interface. The top bar indicates the workspace is 'API Network'. The main area shows a GET request to 'https://localhost:4000/api/loansledger'. The 'Send' button is highlighted. Below the request, the 'Query Params' section is visible, showing a table with columns 'Key', 'Value', and 'Description'. The 'Test Results' tab is selected, showing a list of 5 test results, all of which are 'PASSED'. The status bar at the bottom indicates a '200 OK' response with a response time of 12 ms and a size of 2.47 KB.

GET https://localhost:4000/api/loansledger

Params Authorization Headers (8) Body Scripts Settings

Query Params

Key	Value	Description
username	testuser	
password	password123	
Key	Value	Description

Body Cookies Headers (7) Test Results (5/5)

200 OK · 12 ms · 2.47 KB

Testing your API endpoints one by one? Run your entire collection and test end-to-end workflows effortlessly. [Learn more](#)

Run Collection

Filter Results

- PASSED Response status code is 200
- PASSED Response time is less than 200ms
- PASSED Response must have the required Content-Type header
- PASSED Response body is not empty
- PASSED Response body is in valid HTML format

*Note: We can also see that our custom middleware is logging the api requests – showing that they were made successfully without error.*

The screenshot shows a Visual Studio Code editor window titled "Module 7 Project Assignment". The Explorer sidebar on the left displays the project structure, including files like `config.env`, `certs`, `controllers`, `custom-middleware`, `dev-data`, `models`, `node_modules`, `public`, `routes`, `views`, `bookRoutes.js`, `userRoutes.js`, `viewRoutes.js`, `bookExchangeForm...`, `bookExchange...`, `bookExchangeAdd...`, `bookExchangeDet...`, `bookListForm.ejs`, `images`, `login`, `appError.ejs`, `home.ejs`, `X-UnitTests`, `app.js`, `config.env`, `Dockerfile`, `package-lock.json`, `package.json`, and `server.js`. The `config.env` file is open in the editor, showing the following configuration:

```
1 NODE_ENV=development
2 PORT=4000
3 DATABASE=mongodb+srv://btrinkle52:lvrpGisucpwqkkn@trinklecluster.obqkf.mongodb.net/?retrywrites=true&w=majority&appName=Trinklecluster
4 DATABASE_PASSWORD=lvrpGisucpwqkkn
5 API_VERSION=/api/v1
6 JWT_SECRET=the-quick-brown-fox-jumped-over-the-happy-developer
7 JWT_EXPIRES_IN=10d
8 JWT_COOKIE_EXPIRES_IN=9000
```

The TERMINAL panel at the bottom shows the output of a REST client, displaying various HTTP requests and responses, including GET requests for `/`, `/img/logo.png`, `/img/users/We.jpg`, `/books`, `/books/newBookForm`, `/protected`, `/api/books`, `/api/books/username=testuser&password=password123`, and `/api/loansledger`.