

**Module 8: Final Project****Submitted By: Praveen Kumar Siddela****IFT 458 & IFT 554: Middleware Programming & Database Security****Dinesh Sthapit****Due Date: 04/26/2024**

## Introduction

### Overview of the project

The project's objective is to create a Book Exchange Application facilitating users to list and exchange books seamlessly with their peers. It's designed as a lightweight web application prioritizing smooth user experiences for registration, password management, and book listing. Following the MVC (Model-View-Controller) pattern in its architecture ensures scalability and maintainability by separating concerns effectively.

### Problem Statement

Conventional book exchange methods are inefficient and inaccessible, often relying on physical meetings or complex online platforms. This initiative aims to overcome these shortcomings by offering a digital platform where users can effortlessly list their books, explore available titles, and initiate exchanges with others. The aim is to simplify the book exchange process, making it convenient and accessible to a broader audience.

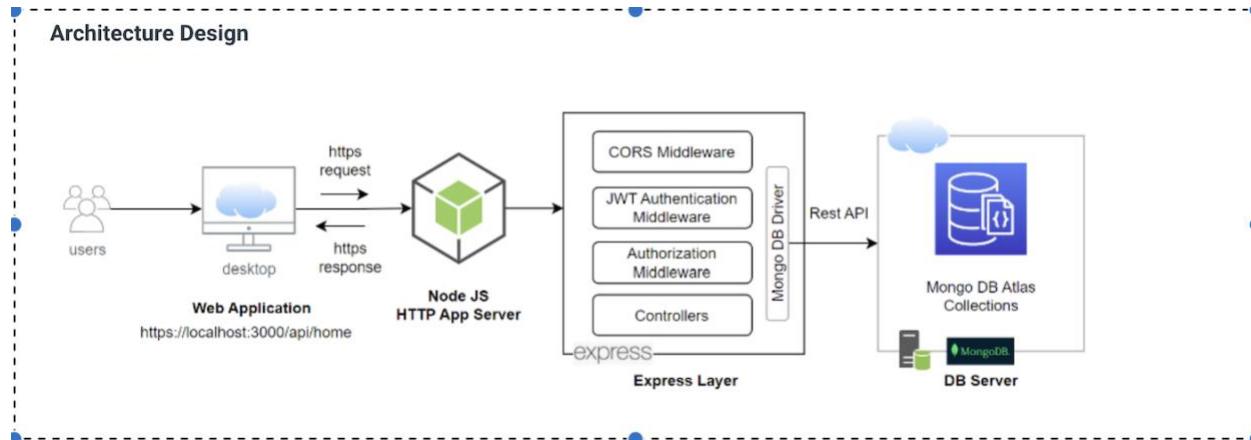
### Scope and Limitations

The project encompasses developing a web-based application featuring user authentication, book listing, and exchange functionalities. Users can register, log in, list their books, browse titles, and initiate exchanges. However, the application doesn't handle physical book shipments or payments; it solely facilitates the exchange process by connecting users based on their preferences.

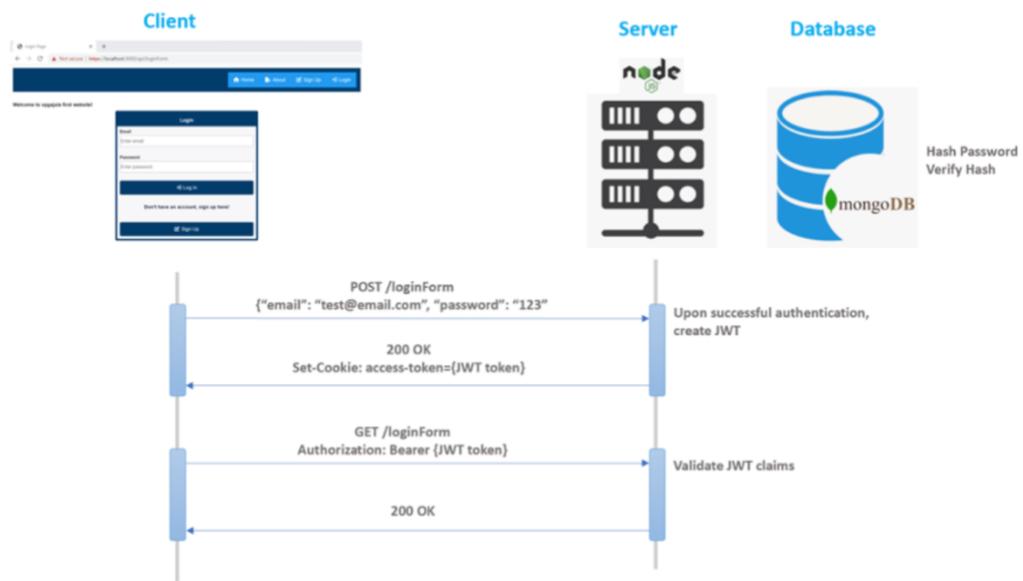
### Justification of the Technology Stack Used

The technology stack comprises Node.js for server-side development, Express.js for constructing the web application framework, MongoDB for efficient data storage, and EJS (Embedded JavaScript) for server-side templating. These technologies were chosen for their reliability, scalability, and user-friendly nature in web application development. Node.js and Express.js offer a lightweight and effective server environment, while MongoDB ensures flexibility and scalability in storing user and book data. EJS enables dynamic content rendering on the server side, enhancing user interaction.

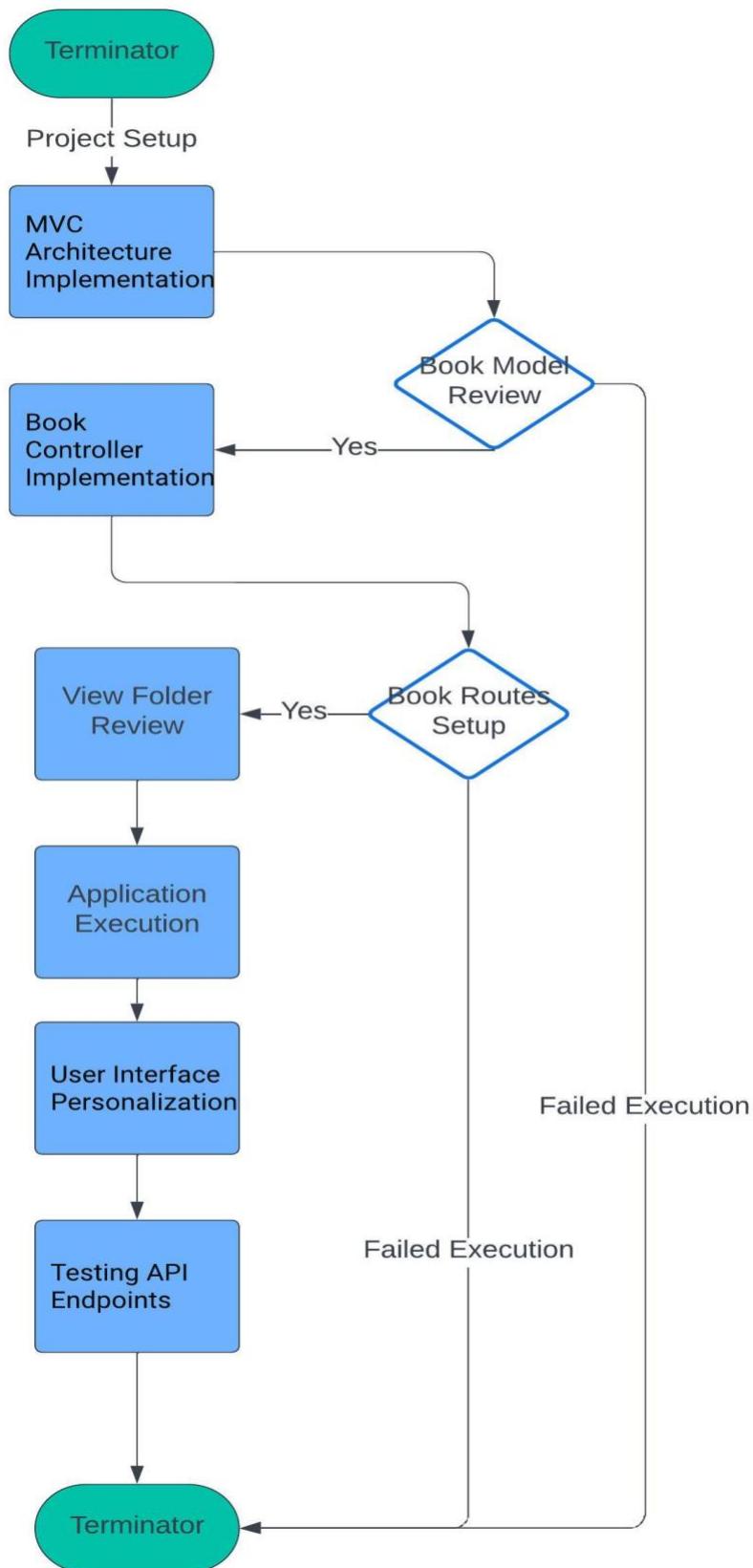
## Architecture Diagrams



## Security Flow:



## Flowchart



**Pseudocode:****// Server-side Process:**

1. Respond to incoming requests along specified routes.
2. Determine the appropriate controller function for each request.
3. Handle data processing and interaction with the model.
4. Formulate and dispatch responses back to the client.

**// Model (Data Management):**

1. Establish structured schemas for book and user data representation.
2. Develop methods to execute database operations, including creation, retrieval, updating, and deletion.
3. Implement procedures for data validation and manipulation.
4. Provide processed data to the controller for further handling.

**// Client-side Actions:**

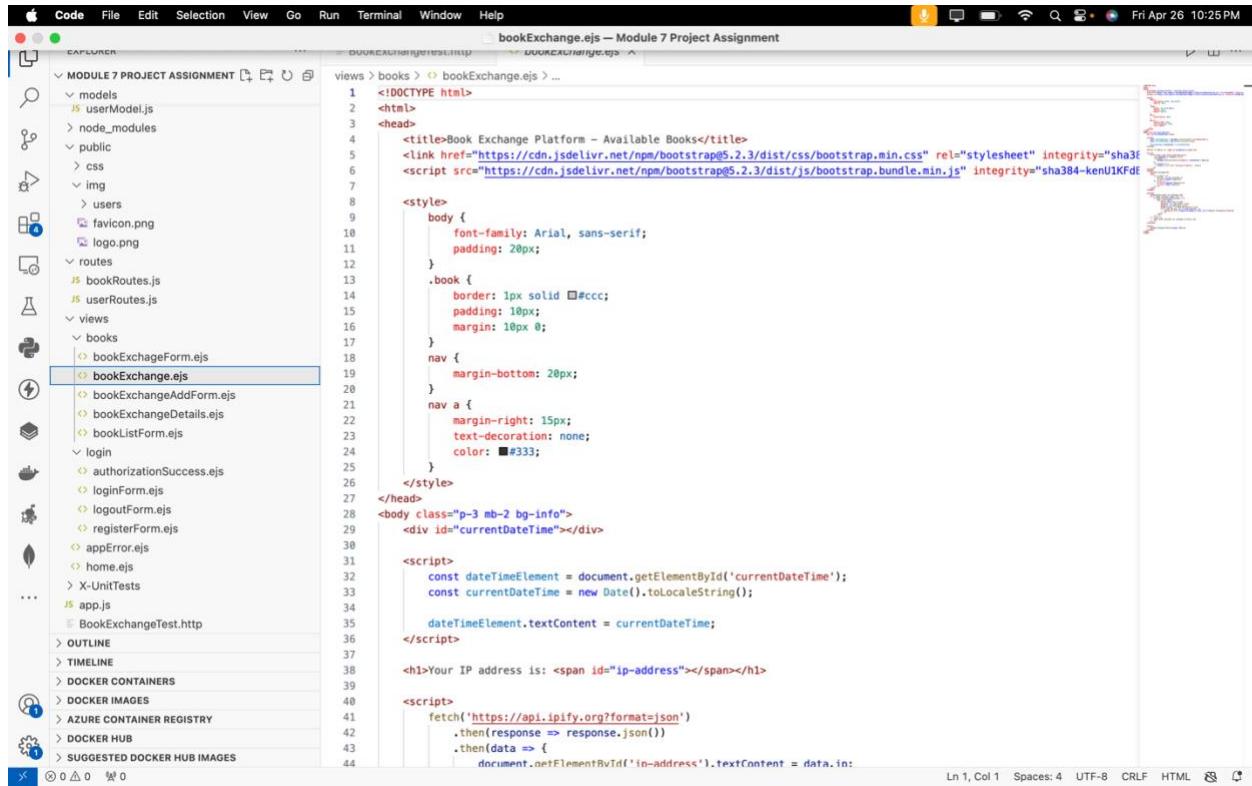
1. Dynamically construct user interface components based on application state.
2. Monitor and respond to user interactions, like form submissions and button clicks.
3. Initiate relevant actions, such as sending requests to the server, upon user interaction.
4. Reflect updates to the interface in accordance with server responses.

## Code Snippets:

## bookExchangeForm.ejs

- 1. Form Submission:** The `<form>` element defines a form for submitting book details to the server. It specifies the `action` attribute as `"/books/add"` for sending the form data to the appropriate endpoint on the server.
  - 2. Input Fields:** Various `<input>` elements are used to capture information about the book, such as title, author, genre, and description. The `required` attribute ensures that these fields must be filled out before the form can be submitted.
  - 3. File Upload:** An `<input type="file">` element allows users to upload a book cover image along with the other details.

## bookExchange.ejs



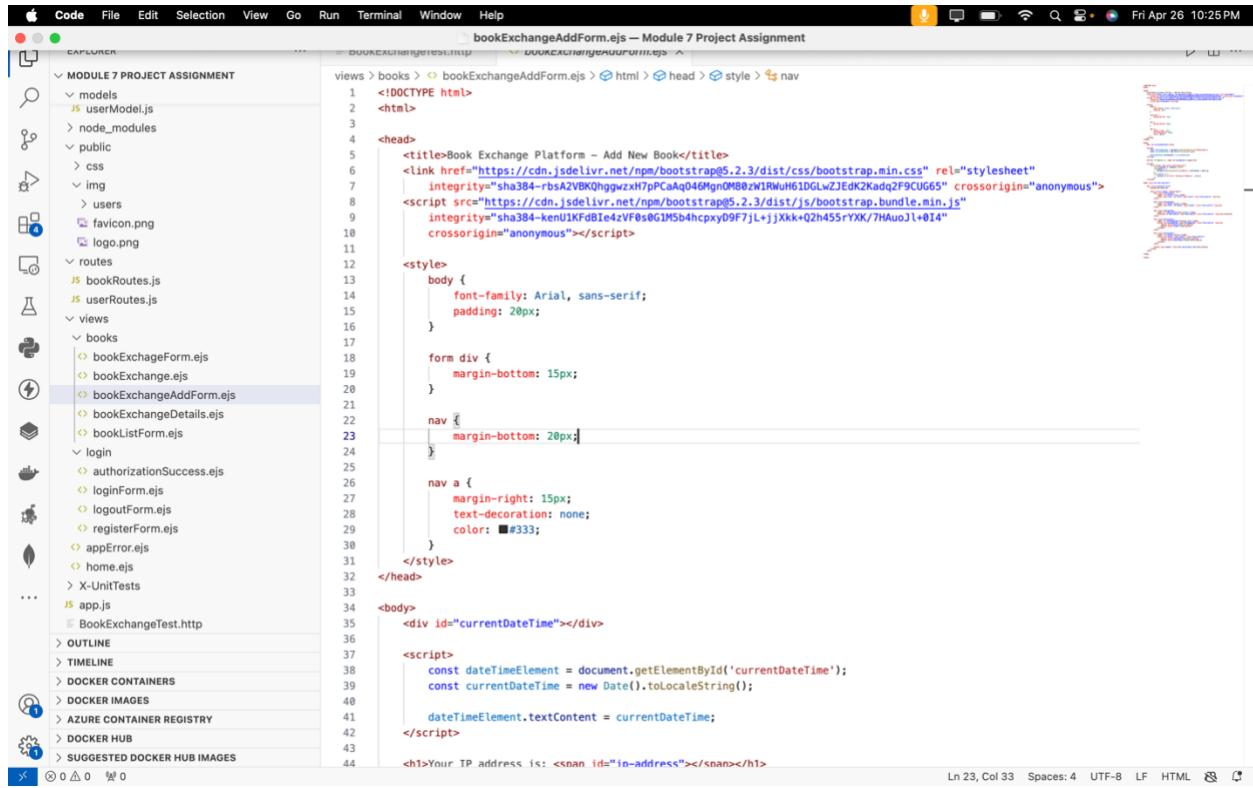
```

<!DOCTYPE html>
<html>
  <head>
    <title>Book Exchange Platform - Available Books</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-...
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-...
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
    }
    .book {
      border: 1px solid #ccc;
      padding: 10px;
      margin: 10px 0;
    }
    nav {
      margin-bottom: 20px;
    }
    nav a {
      margin-right: 15px;
      text-decoration: none;
      color: #333;
    }
  </style>
</head>
<body class="p-3 mb-2 bg-info">
  <div id="currentDateTime"></div>
  <script>
    const dateTimeElement = document.getElementById('currentDateTime');
    const currentDate = new Date().toLocaleString();
    dateTimeElement.textContent = currentDate;
  </script>
  <h1>Your IP address is: <span id="ip-address"></span></h1>
  <script>
    fetch('https://api.ipify.org?format=json')
      .then(response => response.json())
      .then(data => {
        document.getElementById('ip-address').textContent = data.ip;
      })
  </script>
</body>

```

- Error Handling:** The Fetch API is used to asynchronously fetch data from an external API (<https://api.ipify.org?format=json>). Error handling is implemented to catch and log any errors that may occur during the fetch operation.
- Footer:** The <footer> section provides copyright information for the Book Exchange Platform, ensuring proper attribution and ownership acknowledgment.

## bookExchangeAddForm.ejs



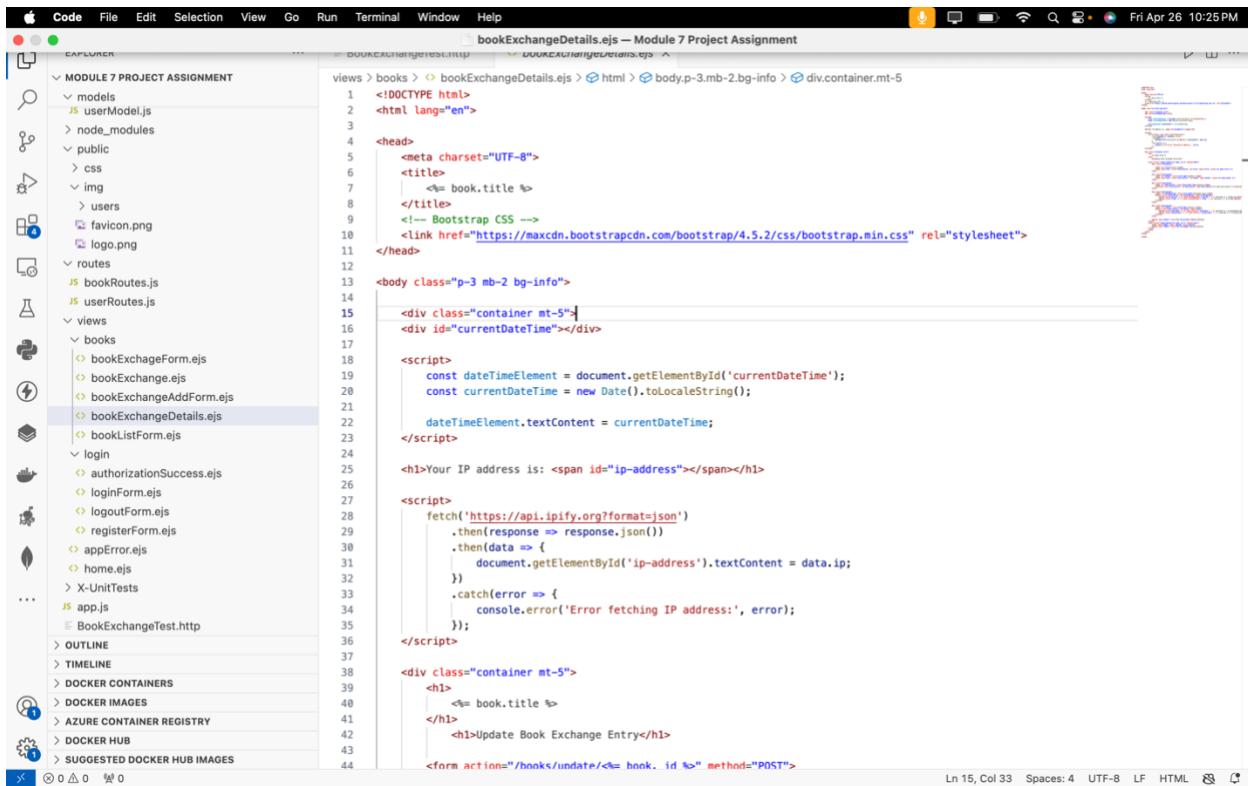
```

 1  <!DOCTYPE html>
 2  <html>
 3
 4  <head>
 5    <title>Book Exchange Platform - Add New Book</title>
 6    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet"
 7      integrity="sha384-+Qq4yRktrR/wBzKQgqwxH7pCaQ046Mg0#B0z1RNuH61DGwZ3EdK2adg2F9CU65" crossorigin="anonymous">
 8    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
 9      integrity="sha384-+ken1KfdB1e4zF0s@G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+@14"
10      crossorigin="anonymous"></script>
11
12  <style>
13    body {
14      font-family: Arial, sans-serif;
15      padding: 20px;
16    }
17
18    form div {
19      margin-bottom: 15px;
20    }
21
22    nav {
23      margin-bottom: 20px;
24    }
25
26    nav a {
27      margin-right: 15px;
28      text-decoration: none;
29      color: #333;
30    }
31  </style>
32
33  </head>
34
35  <body>
36    <div id="currentDateTime"></div>
37
38    <script>
39      const dateTimeElement = document.getElementById('currentDateTime');
40      const currentDateTime = new Date().toLocaleString();
41
42      dateTimeElement.textContent = currentDateTime;
43    </script>
44
45    <h1>Your IP address is: <span id="ip-address"></span></h1>

```

- External Dependencies:** This is achieved through the `<link>` and `<script>` tags in the `<head>` section.
- Dynamic Content:** JavaScript embedded within the HTML document retrieves and displays the current date and time, as well as the user's IP address, using the Fetch API. This dynamic content is injected into specific elements within the `<body>` section.
- Form Submission:** The `<form>` element defines a form for adding a new book. It specifies the action attribute as `"/books"` and the method attribute as `"post"`, indicating that form data will be sent to the `"/books"` endpoint using the HTTP POST method upon submission.

## bookExchangeDetails.ejs



```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <title>
7          | <%= book.title %>
8      </title>
9      <!-- Bootstrap CSS -->
10     <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
11 </head>
12
13 <body class="p-3 mb-2 bg-info">
14
15     <div class="container mt-5">
16         <div id="currentDateTime"></div>
17
18     <script>
19         const dateTimeElement = document.getElementById('currentDateTime');
20         const currentDateTime = new Date().toLocaleString();
21
22         dateTimeElement.textContent = currentDateTime;
23     </script>
24
25     <h1>Your IP address is: <span id="ip-address"></span></h1>
26
27     <script>
28         fetch('https://api.ipify.org?format=json')
29             .then(response => response.json())
30             .then(data => {
31                 document.getElementById('ip-address').textContent = data.ip;
32             })
33             .catch(error => {
34                 console.error('Error fetching IP address:', error);
35             });
36     </script>
37
38     <div class="container mt-5">
39         <h1>
40             | <%= book.title %>
41         </h1>
42             <h1>Update Book Exchange Entry</h1>
43
44         <form action="/books/update/<%= book.id %>" method="POST">

```

Ln 15, Col 33 Spaces: 4 UTF-8 LF HTML

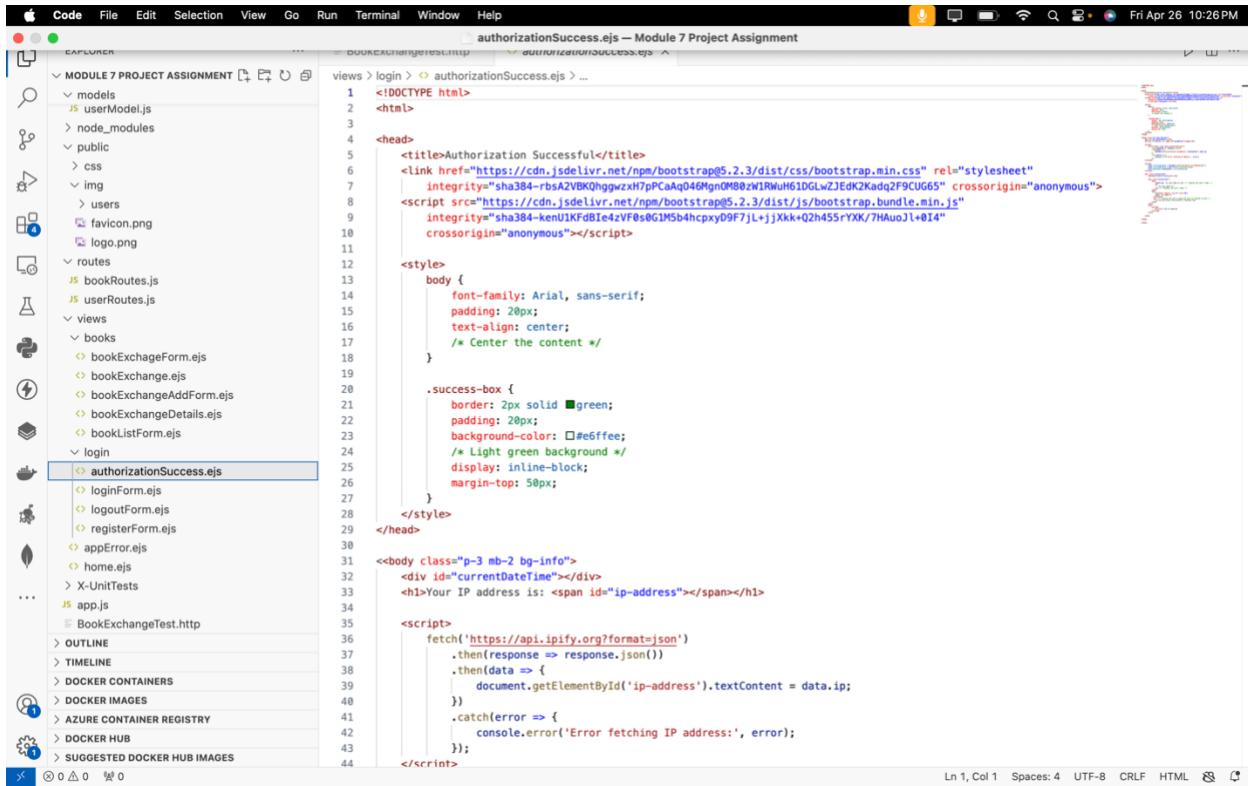
## bookListForm.ejs



The screenshot shows a Java IDE interface with the following details:

- Project Explorer (Left):** Shows the project structure for "MODULE 7 PROJECT ASSIGNMENT". It includes "models", "public" (with "css" and "img" subfolders), "routes" (with "bookRoutes.js" and "userRoutes.js" files), and "views" (with "books" (containing "bookExchangeForm.ejs", "bookExchange.ejs", "bookExchangeAddForm.ejs", "bookExchangeDetails.ejs", and "bookListForm.ejs"), "login" (containing "authorizationSuccess.ejs", "loginForm.ejs", "logoutForm.ejs", "registerForm.ejs", and "appError.ejs"), and "home.ejs"), "X-UnitTests", and "app.js" (with "BookExchangeTest.http" as a child).
- Code Editor (Center):** Displays the content of "bookListForm.ejs". The code is a JSP page with embedded Java code. It includes imports for JSTL and Java's Date class. The page uses Bootstrap 5.2.3 for styling. It features a navigation bar with links to "home", "books", and "login". The main content area displays a list of books with a "View Details" button for each.
- Terminal (Bottom Left):** Shows the command "mvn clean package" being run in the terminal.
- Output (Bottom Right):** Displays the build logs, including the deployment of the application to a local server.

## authorizationSuccess.ejs



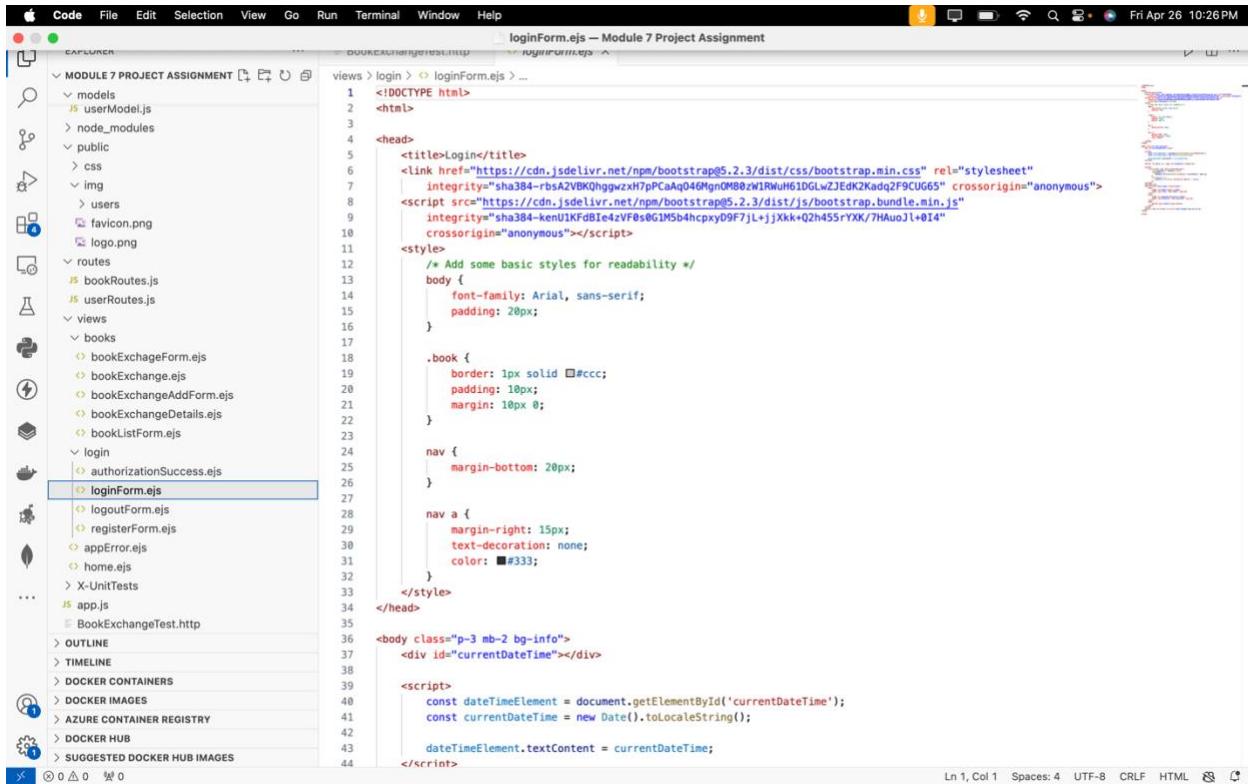
```

views > login > authorizationSuccess.ejs > ...
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5    <title>Authorization Successful</title>
6    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet"
7      integrity="sha384-rbsA2VbkH7H7pCaaQ046Mg0#80z1RnUuH61DGwZ3EdK2Kadg2F9CU65" crossorigin="anonymous">
8    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
9      integrity="sha384-ken1UKfDlE4zV#s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4"
10     crossorigin="anonymous"></script>
11
12  <style>
13    body {
14      font-family: Arial, sans-serif;
15      padding: 20px;
16      text-align: center;
17      /* Center the content */
18    }
19
20    .success-box {
21      border: 2px solid #green;
22      padding: 20px;
23      background-color: ##6fee;
24      /* Light green background */
25      display: inline-block;
26      margin-top: 50px;
27    }
28  </style>
29  </head>
30
31  <body class="p-3 mb-2 bg-info">
32    <div id="currentDateTime"></div>
33    <h1>Your IP address is: <span id="ip-address"></span></h1>
34
35  <script>
36    fetch('https://api.ipify.org?format=json')
37      .then(response => response.json())
38      .then(data => {
39        document.getElementById('ip-address').textContent = data.ip;
40      })
41      .catch(error => {
42        console.error('Error fetching IP address:', error);
43      });
44  </script>

```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF HTML

## loginForm.ejs



```

Code File Edit Selection View Go Run Terminal Window Help
loginForm.ejs — Module 7 Project Assignment
views > login > loginForm.ejs ...
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <title>Login</title>
6   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet"
7     integrity="sha384-rbsA2ElrTCq2CHQqKo0iPjvFqZo8rMyXqT8q0dHwvZ0bqkvj0Pd0Q==" crossorigin="anonymous">
8   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
9     integrity="sha384-ken1KfdB1e4zVF0s0G1M5b4hpxyD9F7jL+jXkk+Q2h455rYXK/7HAuoJl+0I4"
10    crossorigin="anonymous"></script>
11
12   /* Add some basic styles for readability */
13   body {
14     font-family: Arial, sans-serif;
15     padding: 20px;
16   }
17
18   .book {
19     border: 1px solid #ccc;
20     padding: 10px;
21     margin: 10px 0;
22   }
23
24   nav {
25     margin-bottom: 20px;
26   }
27
28   nav a {
29     margin-right: 15px;
30     text-decoration: none;
31     color: #333;
32   }
33
34 </style>
35
36 <body class="p-3 mb-2 bg-info">
37   <div id="currentDateTime"></div>
38
39 <script>
40   const dateTimeElement = document.getElementById('currentDateTime');
41   const currentDateTime = new Date().toLocaleString();
42
43   dateTimeElement.textContent = currentDateTime;
44 </script>

```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF HTML

## logoutForm.ejs



The screenshot shows a Java IDE interface with the following details:

- Toolbar:** Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help.
- Top Status Bar:** logoutForm.ejs — Module 7 Project Assignment, Fri Apr 26 10:26PM.
- Left Sidebar (Project Explorer):** Shows the project structure for "MODULE 7 PROJECT ASSIGNMENT".
- Central Area:** Code editor for "logoutForm.ejs". The code is an EJS template with the following content:

```
<!DOCTYPE html>
<html>
<head>
  <title>Logout</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gf9QF4dGWUDtY�9Pmbq4" data-bbox="350 100 880 150"/>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFd" data-bbox="350 150 880 200"/>
</head>
<body class="p-3 mb-2 bg-info">
  <div id="currentDateTime"></div>

  <script>
    const dateTimeElement = document.getElementById('currentDateTime');
    const currentDateTime = new Date().toLocaleString();

    dateTimeElement.textContent = currentDateTime;
  </script>

  <h1>Your IP address is: <span id="ip-address"></span></h1>

  <script>
    fetch('https://api.ipify.org?format=json')
      .then(response => response.json())
      .then(data => {
        document.getElementById('ip-address').textContent = data.ip;
      })
      .catch(error => {
        console.error('Error fetching IP address:', error);
      });
  </script>
<h2>Are you sure you want to logout?</h2>
<form action="<%= api_version %>/users/logout" method="post">
  <button type="submit">Logout</button>
</form>
<p>Want to stay? <a href="#">Go back</a></p>
</body>
</html>
```

The code includes a date/time display, an IP address fetcher, and a logout form.

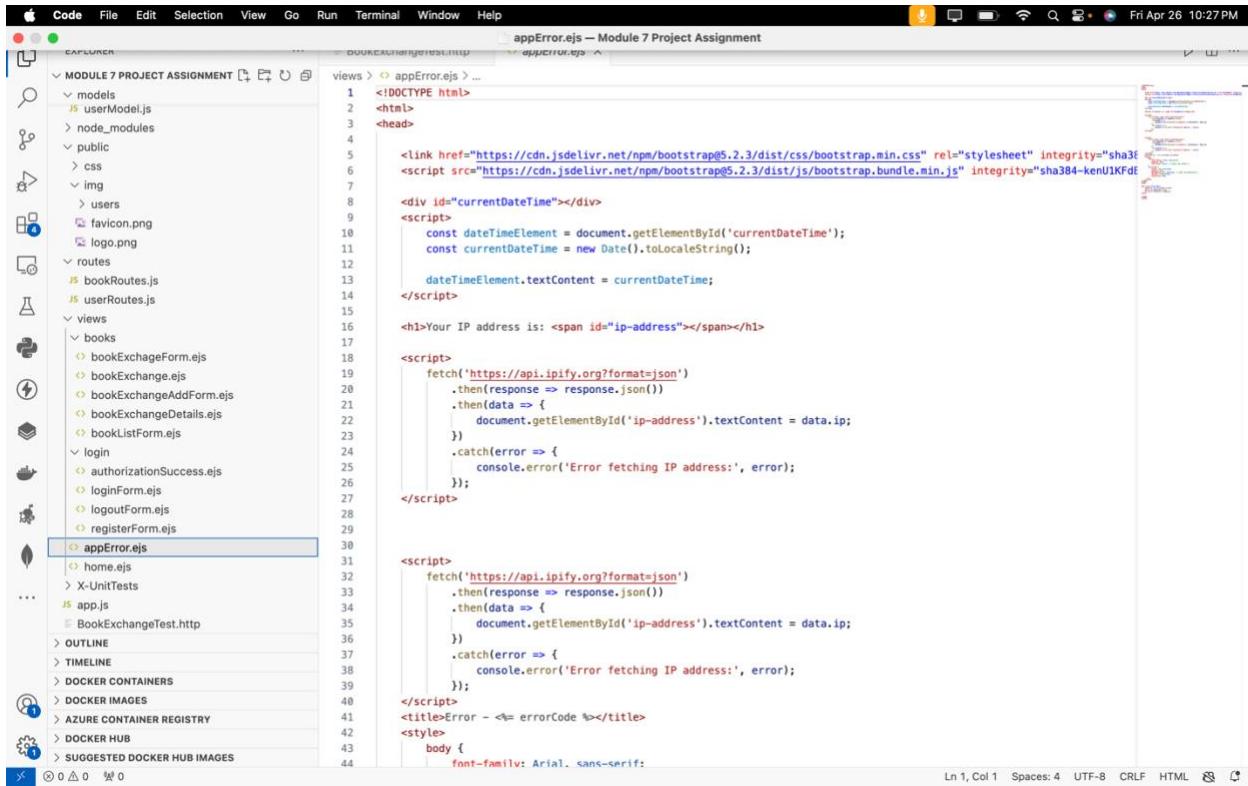
## registerForm.ejs



```
registerForm.ejs — Module 7 Project Assignment
views > login > registerForm.ejs > ...
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <title>Register</title>
6  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2VBRZf2Jp4d7dHxStQfHEzvNPi7Z7070SUsQJHh4IiS5P7hB1gd2kadv29CU65" crossorigin="anonymous">
7  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdEl4zVF0s0GJMSb4hcpxy09F7jl+jjXkk+Q2h455rYXK/7HAu0l+014" crossorigin="anonymous"></script>
8
9  <style>
10     /* Add some basic styles for readability */
11     body {
12         font-family: Arial, sans-serif;
13         padding: 20px;
14     }
15
16     .book {
17         border: 1px solid #ccc;
18         padding: 10px;
19         margin: 10px 0;
20     }
21
22     nav {
23         margin-bottom: 20px;
24     }
25
26     nav a {
27         margin-right: 15px;
28         text-decoration: none;
29         color: #333;
30     }
31
32  </style>
33
34
35  </head>
36
37  <body class="p-3 mb-2 bg-info">
38      <div id="currentDateTime"></div>
39
40  <script>
41      const dateInputElement = document.getElementById('currentDateTime');
42      const currentDate = new Date().toLocaleString();
43
44      dateInputElement.textContent = currentDate;

```

## appError.ejs



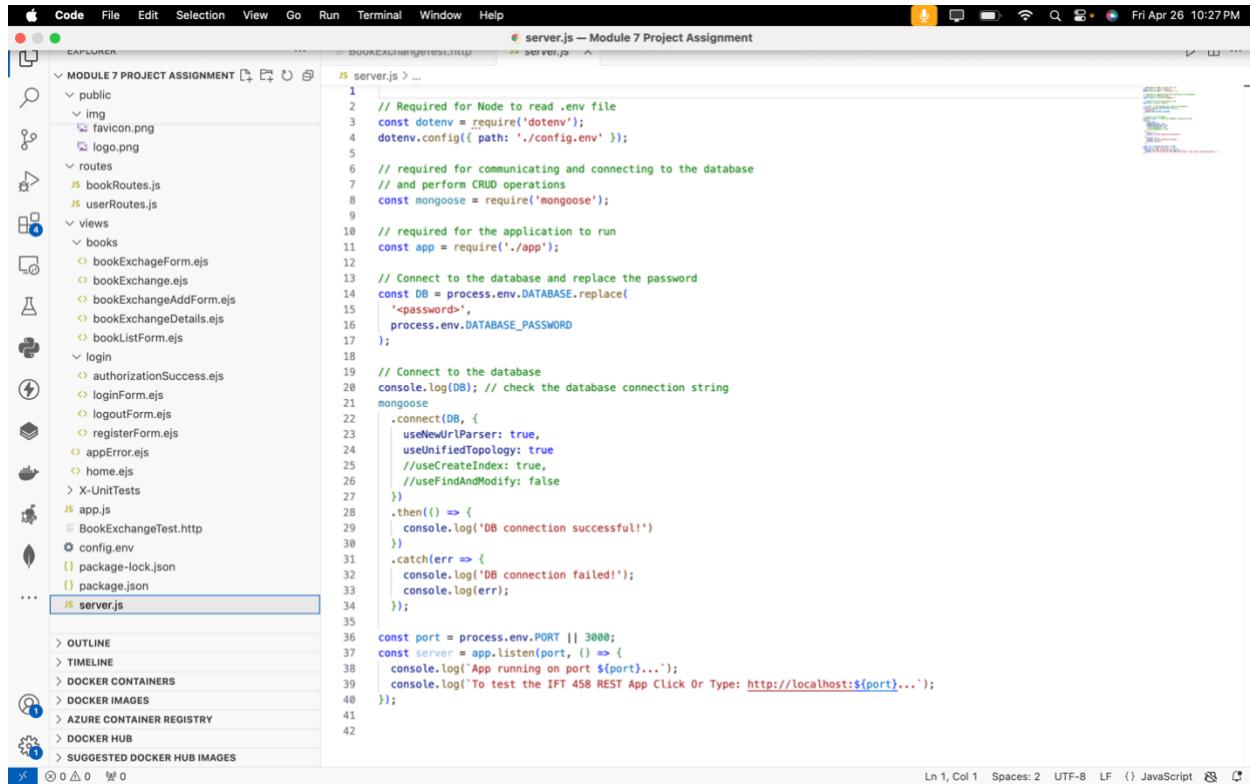
```

1  <!DOCTYPE html>
2  <html>
3  <head>
4
5  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha3F
6  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1Kf
7
8  <div id="currentDateTime"></div>
9  <script>
10 const dateTimeElement = document.getElementById('currentDateTime');
11 const currentDateTime = new Date().toLocaleString();
12
13 dateTimeElement.textContent = currentDateTime;
14 </script>
15
16 <h1>Your IP address is: <span id="ip-address"></span></h1>
17
18 <script>
19   fetch('https://api.ipify.org?format=json')
20     .then(response => response.json())
21     .then(data => {
22       document.getElementById('ip-address').textContent = data.ip;
23     })
24     .catch(error => {
25       console.error('Error fetching IP address:', error);
26     });
27 </script>
28
29
30 <script>
31   fetch('https://api.ipify.org?format=json')
32     .then(response => response.json())
33     .then(data => {
34       document.getElementById('ip-address').textContent = data.ip;
35     })
36     .catch(error => {
37       console.error('Error fetching IP address:', error);
38     });
39 </script>
40 <title>Error - <%= errorCode %</title>
41 <style>
42   body {
43     font-family: Arial, sans-serif;
44

```

home.ejs

## server.js



```

 1 // Required for Node to read .env file
2 const dotenv = require('dotenv');
3 dotenv.config({ path: './config.env' });
4
5 // required for communicating and connecting to the database
6 // and perform CRUD operations
7 const mongoose = require('mongoose');
8
9 // required for the application to run
10 const app = require('./app');
11
12 // Connect to the database and replace the password
13 const DB = process.env.DATABASE.replace(
14   '<password>',
15   process.env.DATABASE_PASSWORD
16 );
17
18 // Connect to the database
19 console.log(DB); // check the database connection string
20 mongoose
21   .connect(DB, {
22     useNewUrlParser: true,
23     useUnifiedTopology: true,
24     //useCreateIndex: true,
25     //useFindAndModify: false
26   })
27   .then(() => {
28     console.log('DB connection successful!')
29   })
30   .catch(err => {
31     console.log('DB connection failed!');
32     console.log(err);
33   });
34
35
36 const port = process.env.PORT || 3000;
37 const server = app.listen(port, () => {
38   console.log(`App running on port ${port}`);
39   console.log(`To test the IFT 458 REST App Click Or Type: http://localhost:\${port}`);
40 });
41
42

```

## API Specifications:

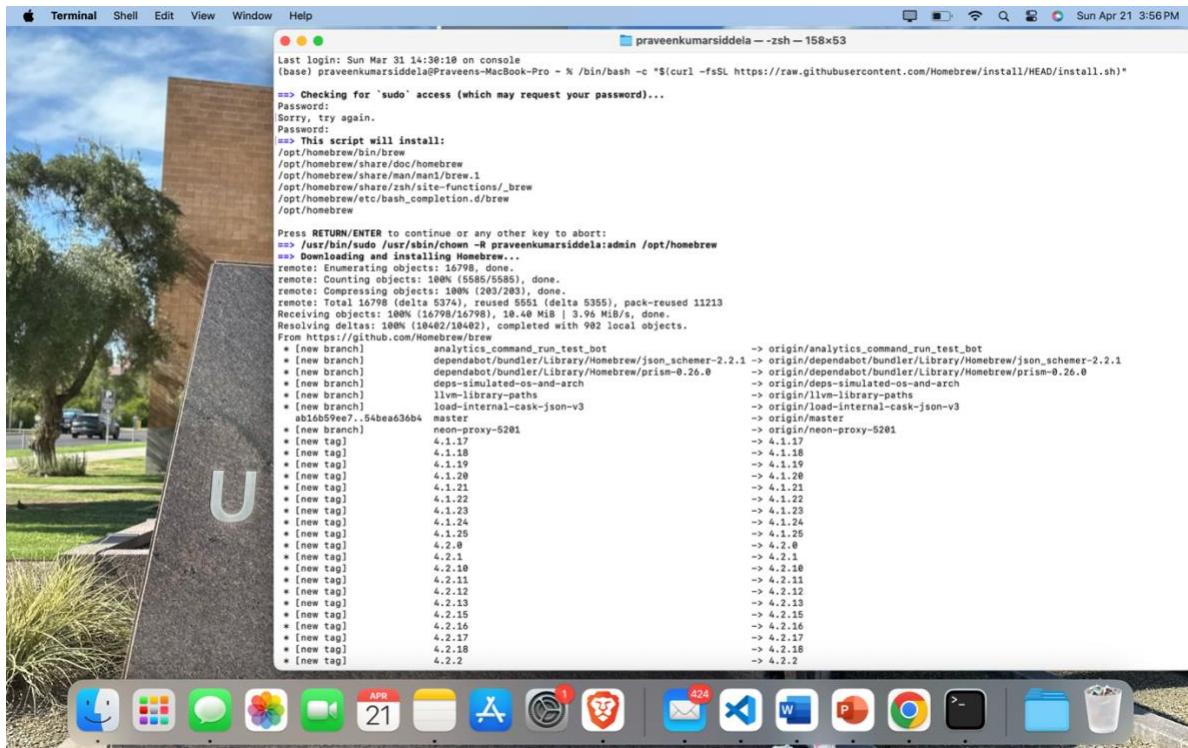
The API specifications detail the endpoints, request methods, and expected formats necessary for communicating with the backend of the application. These specifications follow RESTful principles and encompass functionalities like user authentication, book listing, and exchange.

Specific endpoints include:

- Registering or signing up, accessed via the POST method at <https://localhost:4000/users/signup>
- Logging in, utilizing the POST method at <https://localhost:4000/users/login>
- Accessing protected resources, achieved through the GET method at <https://localhost:4000/books>
- Viewing all available books, accessible via the GET method at <https://localhost:4000/books>
- Adding a book for exchange, done through the POST method at <https://localhost:4000/books/newBookForm>
- Updating or deleting a book exchange, facilitated via the POST method at <http://localhost:4000/books/edit/65e565c188ab31ac49627500>

## Module 7 Activity 1: Generating SSL Certificate and Configuring HTTPS Server in Node.js on Windows

Screenshot 1: Showing the OpenSSL installed confirmation.



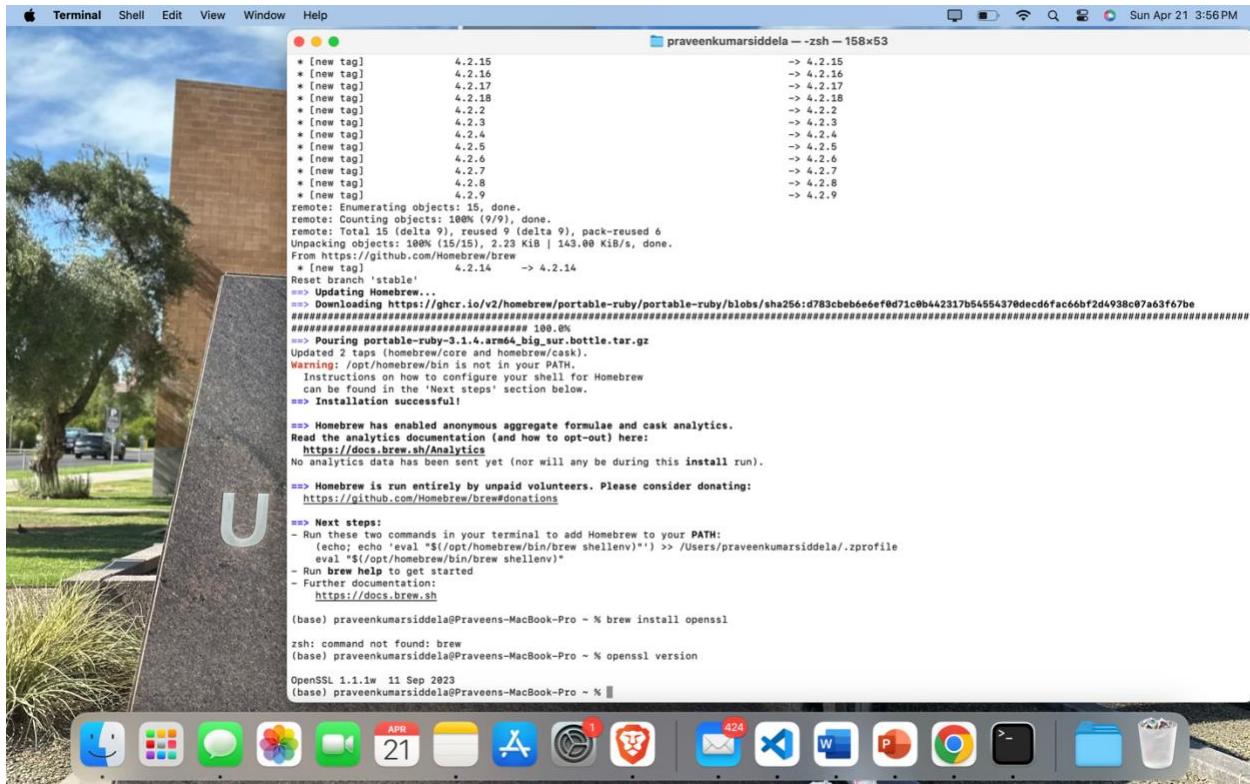
```

Last login: Sun Mar 31 14:38:18 on console
(base) praveenkumarsiddela@Praveen-MacBook-Pro ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

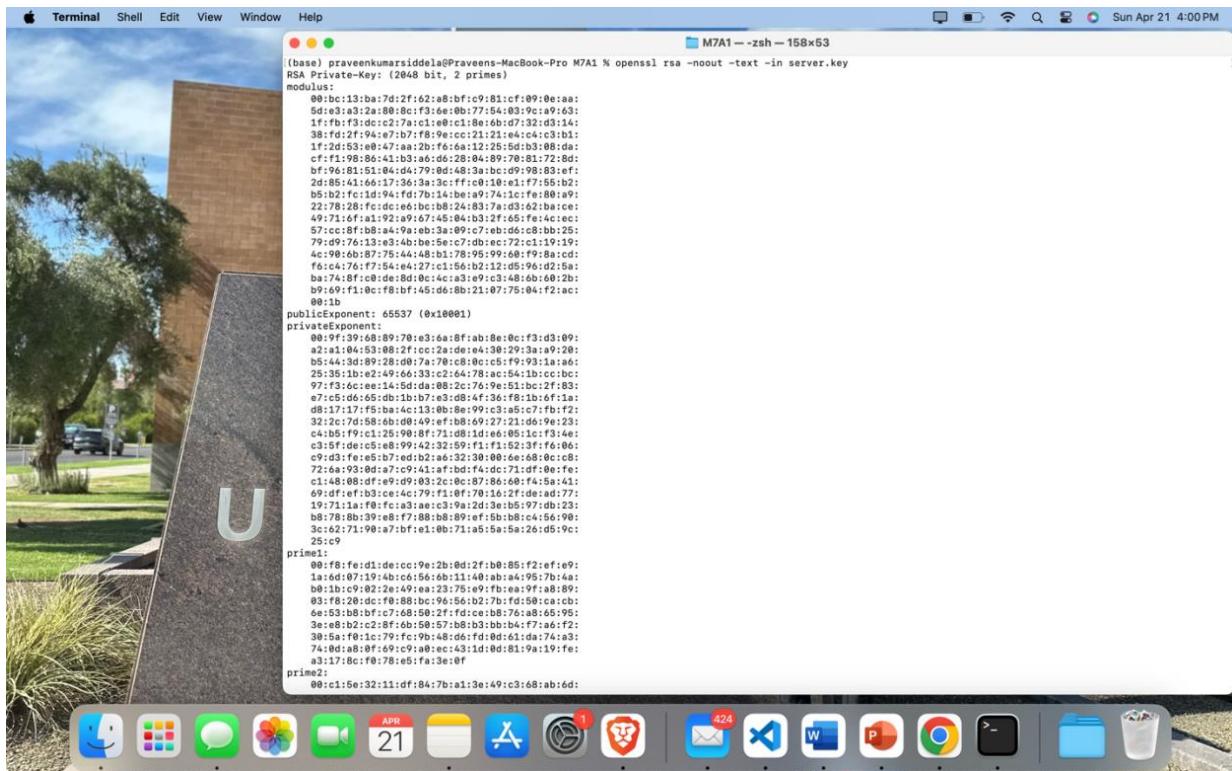
==> Checking for 'sudo' access (which may request your password)...
Password:
Sorry, try again.
Password:
==> This script will install:
/opt/homebrew/bin/brew
/opt/homebrew/share/doc/homebrew
/opt/homebrew/share/man/man1/brew.1
/opt/homebrew/share/zsh/site-functions/_brew
/opt/homebrew/etc/bash_completion.d/brew
/opt/homebrew

Press RETURN/ENTER to continue or any other key to abort:
==> Just press RETURN /user/nv/nv/100 -R praveenkumarsiddela:admin /opt/homebrew
==> Downloading and installing Homebrew...
remote: Enumerating objects: 16798 (5585/5585), done.
remote: Counting objects: 16800 (2693/2693), done.
remote: Compressing objects: 16800 (1260/1260), done.
remote: Total 16798 (delta 5374), reused 5551 (delta 5355), pack-reused 11223
Received objects: 16800 (1260/1260), 18.48 MiB | 3.96 MiB/s, done.
Resolving deltas: 16800 (1042/1042), completed with 902 local objects.
From https://github.com/Homebrew/brew
 * [new branch]         analytics_command_run_test_bot          -> origin/analytics_command_run_test_bot
 * [new branch]         dependabot/bundler/Library/Homebrew/json_schemer-2.2.1 -> origin/dependabot/bundler/Library/Homebrew/json_schemer-2.2.1
 * [new branch]         dependabot/bundler/Library/Homebrew/prism-0.26.0   -> origin/dependabot/bundler/Library/Homebrew/prism-0.26.0
 * [new branch]         dependabot/bundler/Library/Homebrew/prism-0.26.1   -> origin/dependabot/bundler/Library/Homebrew/prism-0.26.1
 * [new branch]         llvm-library-paths                         -> origin/llvm-library-paths
 * [new branch]         load-internal-cask-json-v3                  -> origin/load-internal-cask-json-v3
ab16b59ee7..54bea636be master
 * [new branch]         neon-proxy-5201                            -> origin/neon-proxy-5201
 * [new tag]           4.1.16                                -> 4.1.16
 * [new tag]           4.1.18                                -> 4.1.18
 * [new tag]           4.1.19                                -> 4.1.19
 * [new tag]           4.1.20                                -> 4.1.20
 * [new tag]           4.1.21                                -> 4.1.21
 * [new tag]           4.1.22                                -> 4.1.22
 * [new tag]           4.1.23                                -> 4.1.23
 * [new tag]           4.1.24                                -> 4.1.24
 * [new tag]           4.1.25                                -> 4.1.25
 * [new tag]           4.2.0                                 -> 4.2.0
 * [new tag]           4.2.1                                 -> 4.2.1
 * [new tag]           4.2.10                               -> 4.2.10
 * [new tag]           4.2.11                               -> 4.2.11
 * [new tag]           4.2.12                               -> 4.2.12
 * [new tag]           4.2.13                               -> 4.2.13
 * [new tag]           4.2.15                               -> 4.2.15
 * [new tag]           4.2.16                               -> 4.2.16
 * [new tag]           4.2.17                               -> 4.2.17
 * [new tag]           4.2.18                               -> 4.2.18
 * [new tag]           4.2.2                                -> 4.2.2

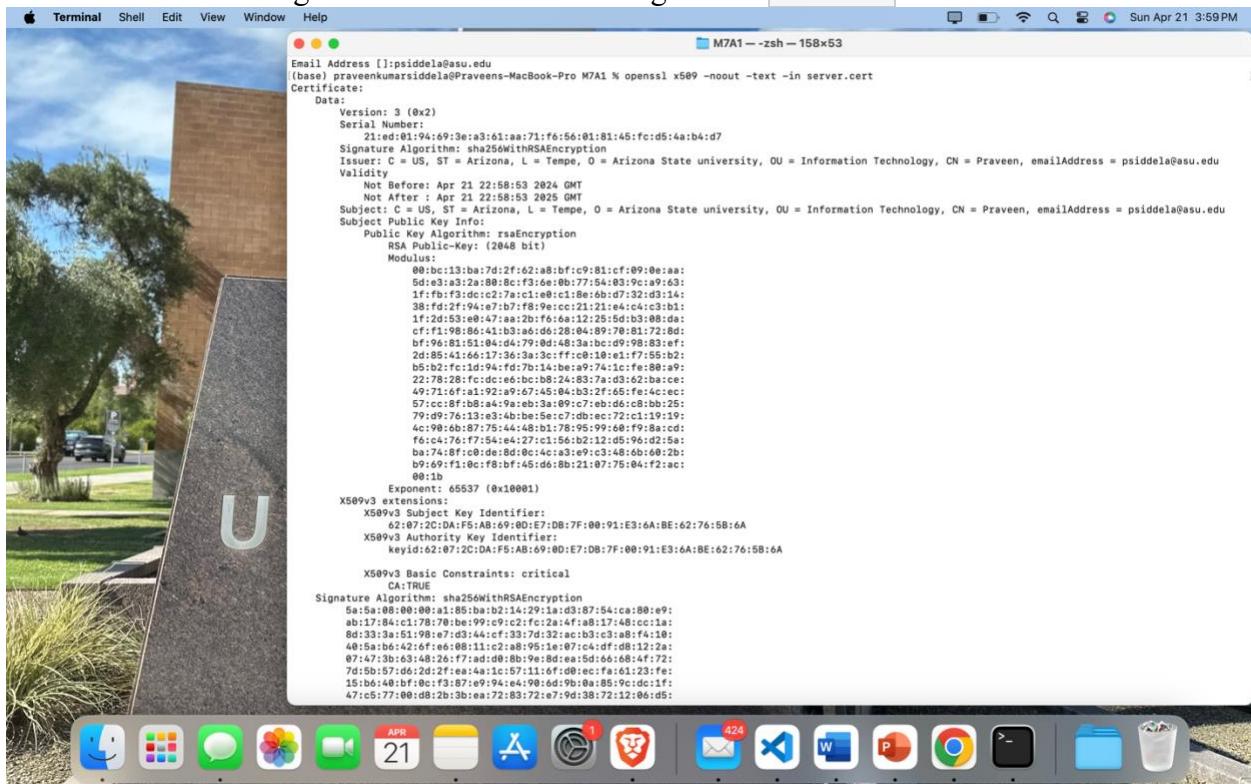
```



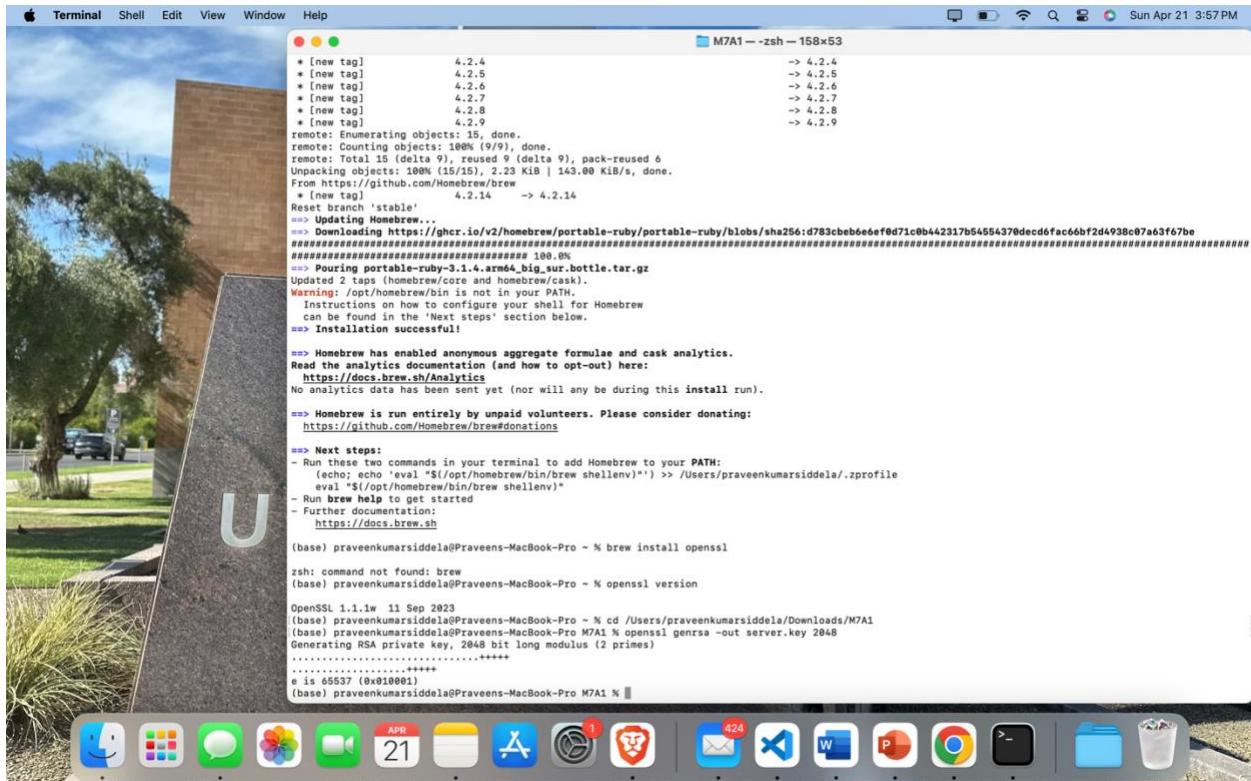
Screenshot 2: Showing Command Prompt after generating server.key.

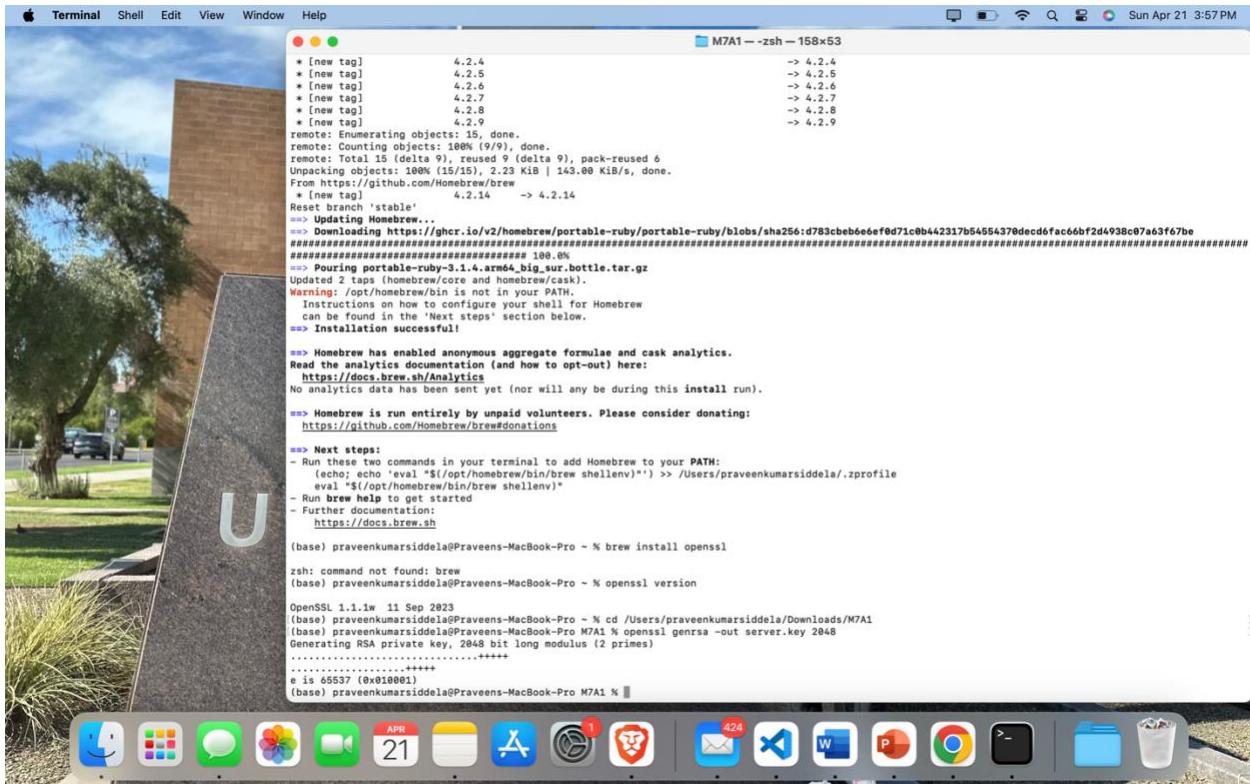


Screenshot 3: Showing execution of command to generate server.cer

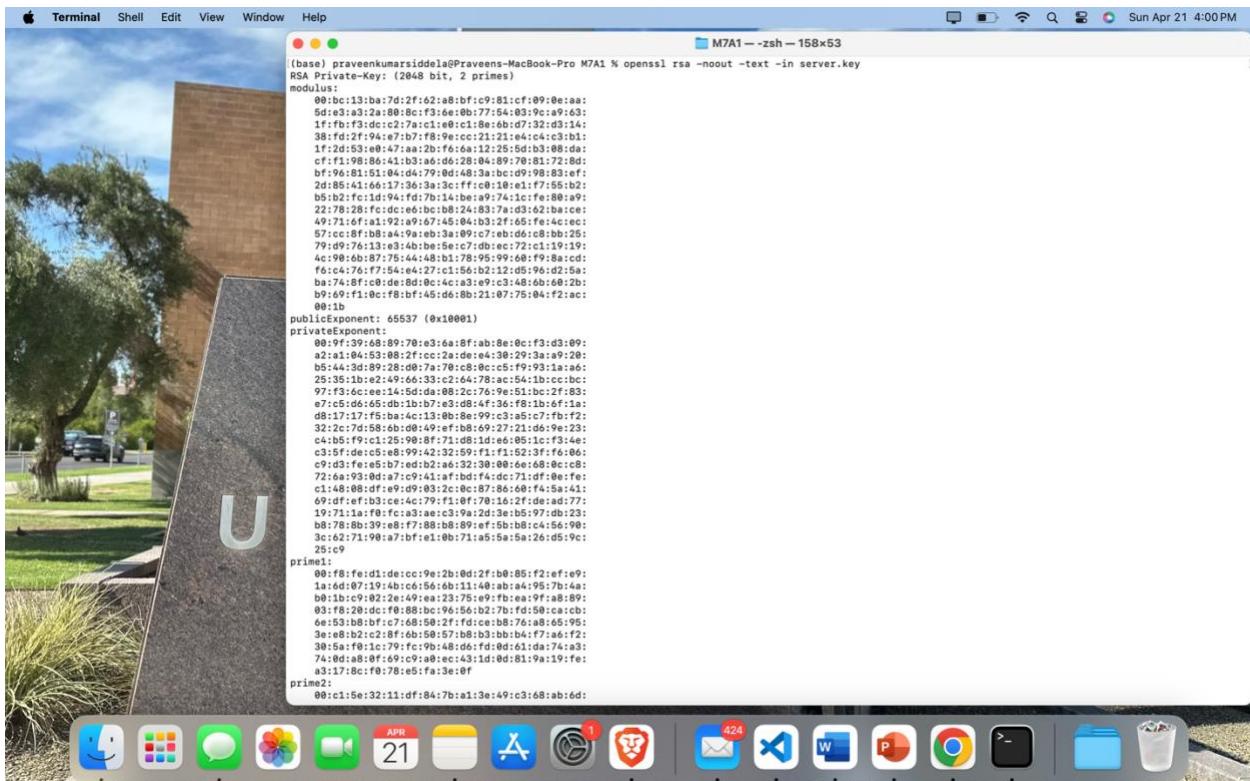


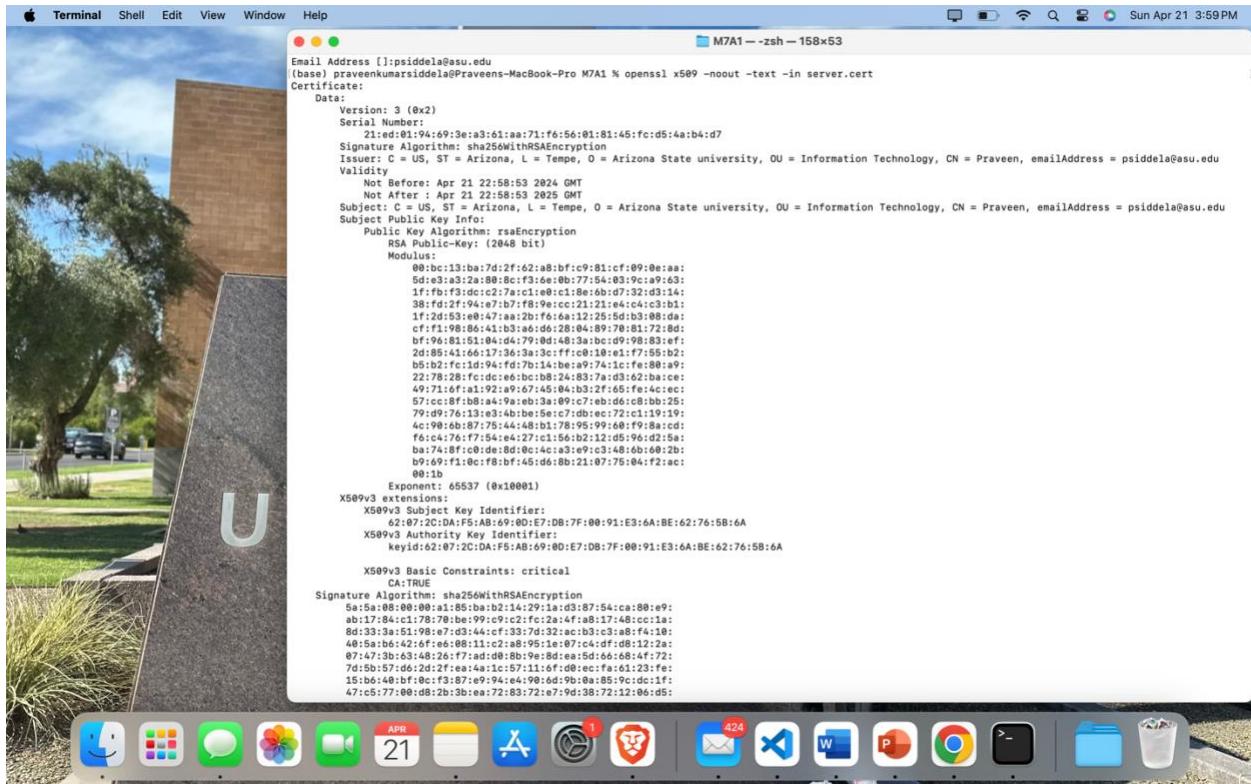
Screenshot 4: Showing filled-in certificate information prompts.





**Screenshot 5:** Showing output of verification commands in Command Prompt.





Screenshot 6: Of the project folder with `server.key` and `server.cert` files.

**server.key**

```

1 -----BEGIN RSA PRIVATE KEY-----
2 MIIEpQIBAAKCAQEaB0f6f59iql/Jgc83Dqpd46MqgIzzbgt3VA0cqMf+/Pcwrb
3 4NGQa9cy0x04/S+U57f4nswhIeTEw7EfLVPgR6o9mo5JV2zCnR8Z1GQ00m11gE
4 1xCbc0z2/lfrNBN5D0U6mNyg8t8hUfMfzY6P/PE0H3VbKsvwdI1P7fL6pdBz+
5 gKkIecJ830a8uC5dEtius51cwh+hkqInR0S2L2x+T0xxz1+4pJr0gnH69bIuyV5
6 2XYT40u+Xsfb7HLBGR1MkGuldR1sX1vNm051s32xh3Q0vnwvaytW0lqGd1/A
7 3o@MTKPPw0b0hYUc5a1fM+19f10shB31UE8qAGw1D0A0B0A0IBAQCf0W1cOnj6u0
8 DPPTCAKBFM1L8wq3u0wK7apILPEVYkoHpwvAzF+2mAp.U11+J2JpcZ2Hs1vBwM
9 vJfzb04Ux0d1lHaeUbwgv+f1mXbG7fj2E82+8tVgtxF/W67BMLpnOpf781s
10 Fvh0Eunu6kn1de1851+e1k19x2B3mBrz1tN13x0n1yf1fx1j28sn7u03
11 7b0kMfAAmpgMyJqk2nyUgvfTcc80/sfTcN/02M5D1eGYPRa0Jwnf777POTHnx
12 D3ANL96t0x/xgv08670m1+zfbl7h41z094141e90JmHWRDX1cZlnv+ElcaVa
13 Wb1vNcXjA0GBApJ+bdMnsLNL7CF8u/pGmHgUvGvNQ0ukUx1k5BvJA1516JN1
14 6fvq6n1J4/g3PC1vJZ2m9vNURl0104/8d0u0/79/rh2qGpuwy909Uf4e5+u0
15 96byMPrwHm8n0W1Qh2n51d2o2nJ0x0DHQ2Bmn+oxM8H1+j4PA0GBANFe
16 MHfHfhuhPkn0dKtt1f4qzINS1w6Cnok7fz10RAu3hBf0hDh5puYSPjMnUttU1
17 QcKuB0U9f9z1t/s251z134pk59v5f0dMC/0yus55t2kQw/UrLyV+LkEes
18 dP10t56te61e-xPMKevA30z1j1eyV4w04AB2k1aG6BALT66m3jZE7NWWBt10
19 Tj/FEajesaaOC18CqkCqMl/jlEscooBjx+f6spIRkt0n61j0gVhG3Q0q+EK73Y
20 tDpYpdyvYJ9J7tir/zs1u0J6w/Hp1055krHEf5T3-nelPf64ZQdGLpBqB1Cpt
21 LnV1qshf3PTf3661HM0nccAoGBA16TPPa0tH)Y0663A3ybkgHwCpUg7P0RfJ5
22 UBW672ts11r140mspQ1Gr711U72x2cjsyDQX38suvrZvQn3+T2Gea65180t8
23 Un1XnK0q7mAg7HDs/xTjP100dt/Eewf0fQG808076Udf1+5PrR+f9X0e1
24 aWUK51kdAoG2Z1L81/g0vcoBn/Fukau144G5UppZtH0b1xMsaoq1uM3/59j2V
25 0TQJ5g4t4A3+yv455ak3q3a3qcd10n7H9Ptu3Nbxowf5g/oec14xLpm1p7719h
26 FVVAxkuuyr4w0p0ka1vZmfkKx0RtLtaoq5j1jg4uFL5fDRhzewd4a
27 -----END RSA PRIVATE KEY-----

```

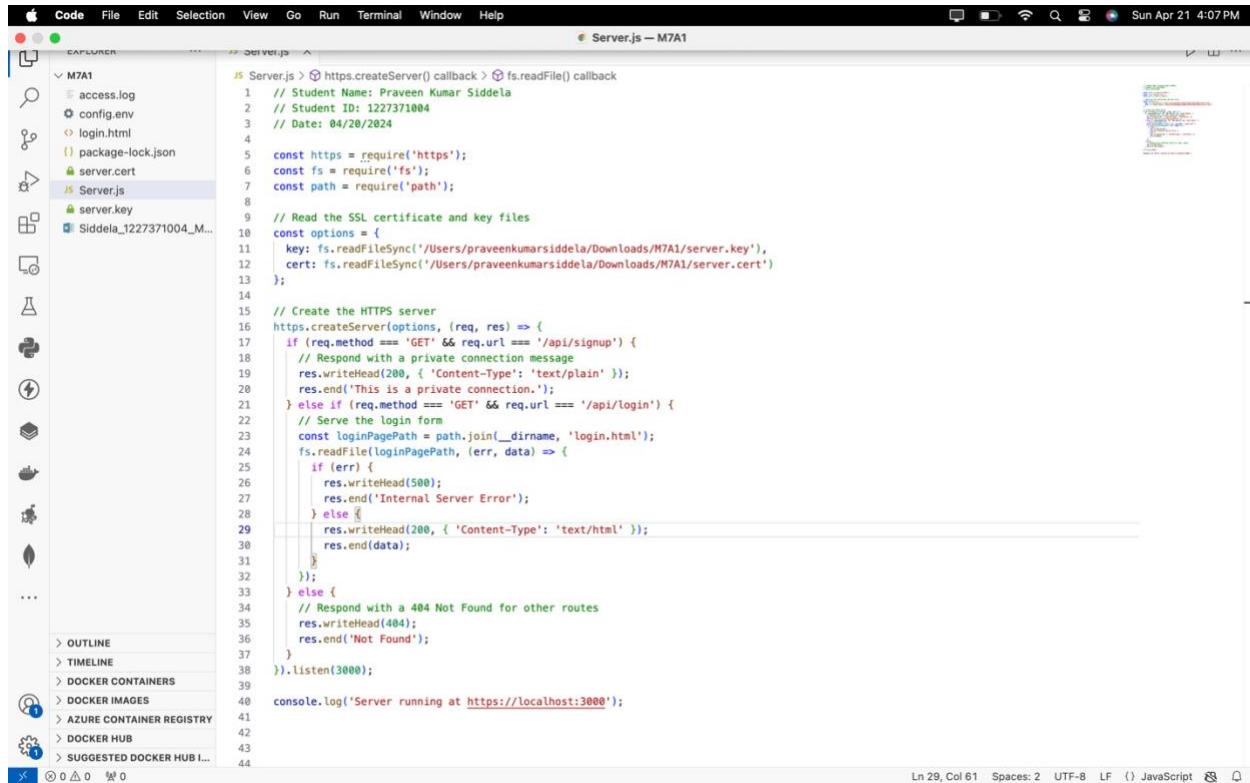
**server.cert**

```

1 -----BEGIN CERTIFICATE-----
2 MIIEzCAxegAwB1gU1e@16k+o2GqcFzWAYFF/NVKTNcw0QYKoZ1hvckNAQEL
3 B0AgwqCzA1BgNVBAYTlTMRAw0gYDQ0IDaDc1m625hM04wAVYDQ0QDAVU
4 ZW1wZTEhMBG8A1UEcwYQJemp9uYSBTdGf0ZS1b1mL2ZJzaxR5Mh8wH0YDQ0Q
5 DB2Jb0Zycm1hG1vb1BUZwNobm9s2d5MRAw0gYDQ0QDadQcmf2ZwVhM8wH0YJ
6 KoZhvcdAQkBFhBwc21kZQgvYU8h3u0z2wIMB4D7D0tMDQyMT1yM7g1M1oX0T11
7 MDQyMT1yNTg1M1owgqaycCAzBgvNBAYTlTMRAw0gYDQ0Q10A0Bcm16b25hM04w
8 DAYDQ0QH0AVUZw1zWtTEhMB8GA1UECwgyQXJpe9uYSBTdGf0ZS1b1mL2ZJzaxR5
9 MR8wH0YDQ0QV0L0B2Jm2v1hgd1UECwgyQXJpe9uYSBTdGf0ZS1b1mL2ZJzaxR5
10 ZWVLMR8wH0YJkoZ1hvNAQkBFhBwc21kZGVgYUBh3uUZwR1M1B1JANBqkgh1G
11 9w0BA0fFA0CA0QABM1BCgkCA0EAwB0f6f59iql/Jgc83Dqpd46RqgIzbg1t3V0c
12 qMf+/Pcm/B4NG0a9cy0x04/S+U57f4nswhIeTew7EfLVPgR6or9mo5Jv2zNcrP
13 8Z1GQ00m1jge1XcBc0z/IoFBNRS0Dg6NwvYg+8tNuFzFy6PP/AE0H3V0K1svwd
14 lP17fL6pdBz+gKkIecJ830a8uC5dEtius51cwh+hkqInR0S2L2x+T0xxz1+4pJr0
15 Ognh69biuy52XYT40u+Xsfb7HLBGR1MkGuldR1sX1vNm051s32xh3Q0vnwvay
16 EtwW0Lq6d1/A3o@MTKPPw0hYUc5a1fM+19f10shB31UE8qWgW1D0A0b01MwTAd
17 BgNwH04EfQ0Vgcs2vra3n238akeNqvnZ2W2oWhYDVR8jB8gwF0UyGcs2vWr
18 a03n238akeNqvnZ2W2oWhYDVR8jB8gwF0UyGcs2vWr
19 AQEAl0iAAChhbgvyCka0d4lyD0pxexXhvvwpnJwwgT6xSwmajTM6U2j1o0TP
20 M30yrlPdqPQQ0FmBHC0kUeB8Tf2B1qB0cF7Y0g963Q156Ng1lmaE9yfVtX
21 110v6KocVxFv026YSP+FbZAvwzhh+0USJ8btmqFnwF78V3ANgr0+pyg3lnnThy
22 EgbVh0@nH0m0ak4gEcEzFjdH60KQ100dc0RHB8eTzEuEl8KUvbgCpBdaw
23 bbtv33dfcEtwyF87Q0wfz/1zlfZtAs6s5Q8YAVIGpsU4rrth+17q1UhaVixw
24 U4MP5uwdahknYx0sV9we2dpZa==
25 -----END CERTIFICATE-----

```

Screenshot 7: Showing the modified Server.js file in your IDE.

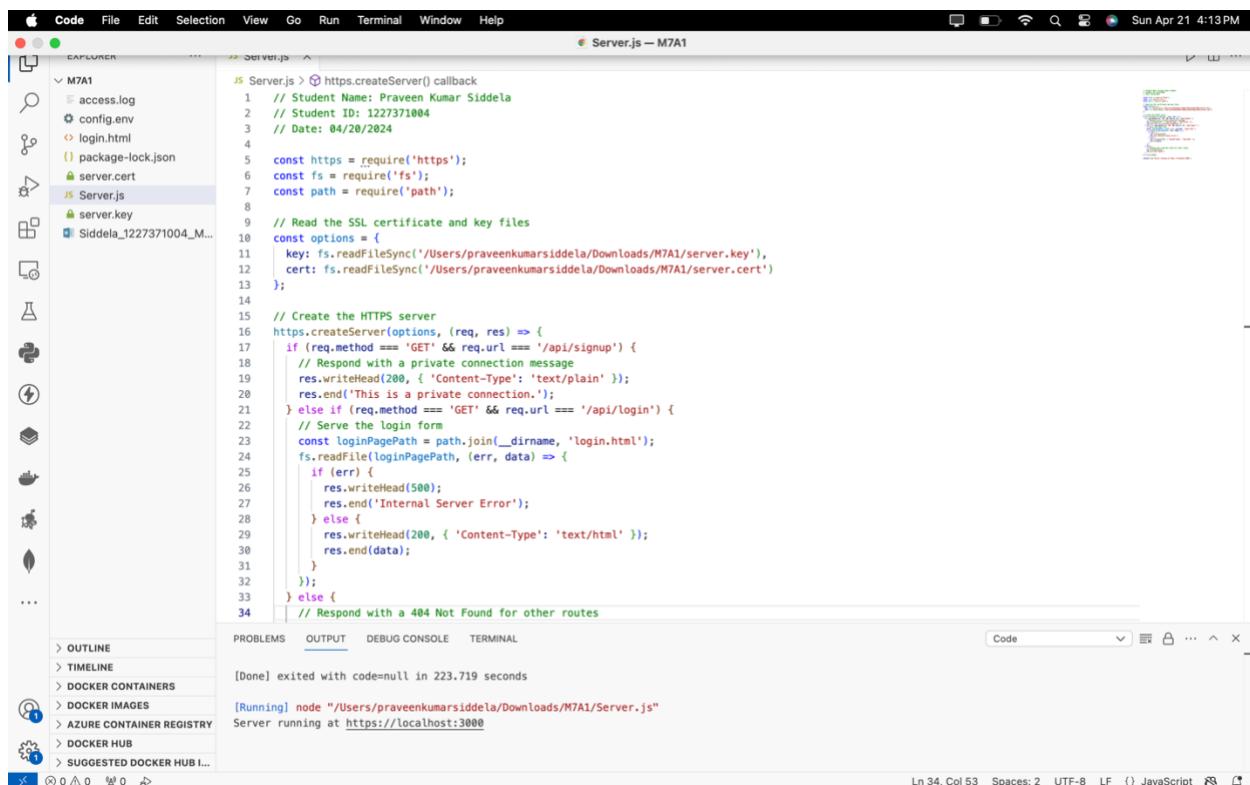


```

JS Server.js > https.createServer() callback > fs.readFile() callback
1 // Student Name: Praveen Kumar Siddela
2 // Student ID: 1227371004
3 // Date: 04/20/2024
4
5 const https = require('https');
6 const fs = require('fs');
7 const path = require('path');
8
9 // Read the SSL certificate and key files
10 const options = {
11   key: fs.readFileSync('/Users/praveenkumarsiddela/Downloads/M7A1/server.key'),
12   cert: fs.readFileSync('/Users/praveenkumarsiddela/Downloads/M7A1/server.cert')
13 };
14
15 // Create the HTTPS server
16 https.createServer(options, (req, res) => {
17   if (req.method === 'GET' && req.url === '/api/signup') {
18     // Respond with a private connection message
19     res.writeHead(200, { 'Content-Type': 'text/plain' });
20     res.end('This is a private connection.');
21   } else if (req.method === 'GET' && req.url === '/api/login') {
22     // Serve the login form
23     const loginPagePath = path.join(__dirname, 'login.html');
24     fs.readFile(loginPagePath, (err, data) => {
25       if (err) {
26         res.writeHead(500);
27         res.end('Internal Server Error');
28       } else {
29         res.writeHead(200, { 'Content-Type': 'text/html' });
30         res.end(data);
31       }
32     });
33   } else {
34     // Respond with a 404 Not Found for other routes
35     res.writeHead(404);
36     res.end('Not Found');
37   }
38 }).listen(3000);
39
40 console.log('Server running at https://localhost:3000');
41
42
43
44

```

Screenshot 8: Showing the terminal with the running Node.js server.



```

JS Server.js > https.createServer() callback > fs.readFile() callback
1 // Student Name: Praveen Kumar Siddela
2 // Student ID: 1227371004
3 // Date: 04/20/2024
4
5 const https = require('https');
6 const fs = require('fs');
7 const path = require('path');
8
9 // Read the SSL certificate and key files
10 const options = {
11   key: fs.readFileSync('/Users/praveenkumarsiddela/Downloads/M7A1/server.key'),
12   cert: fs.readFileSync('/Users/praveenkumarsiddela/Downloads/M7A1/server.cert')
13 };
14
15 // Create the HTTPS server
16 https.createServer(options, (req, res) => {
17   if (req.method === 'GET' && req.url === '/api/signup') {
18     // Respond with a private connection message
19     res.writeHead(200, { 'Content-Type': 'text/plain' });
20     res.end('This is a private connection.');
21   } else if (req.method === 'GET' && req.url === '/api/login') {
22     // Serve the login form
23     const loginPagePath = path.join(__dirname, 'login.html');
24     fs.readFile(loginPagePath, (err, data) => {
25       if (err) {
26         res.writeHead(500);
27         res.end('Internal Server Error');
28       } else {
29         res.writeHead(200, { 'Content-Type': 'text/html' });
30         res.end(data);
31       }
32     });
33   } else {
34     // Respond with a 404 Not Found for other routes
35     res.writeHead(404);
36     res.end('Not Found');
37   }
38 }).listen(3000);
39
40 console.log('Server running at https://localhost:3000');
41
42
43
44

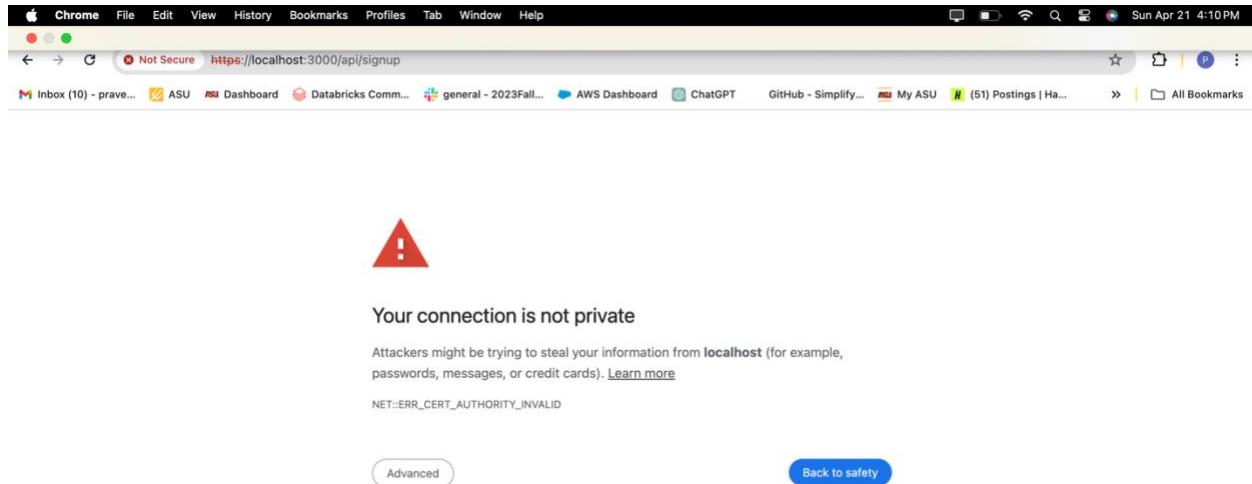
```

[Done] exited with code:null in 223.719 seconds

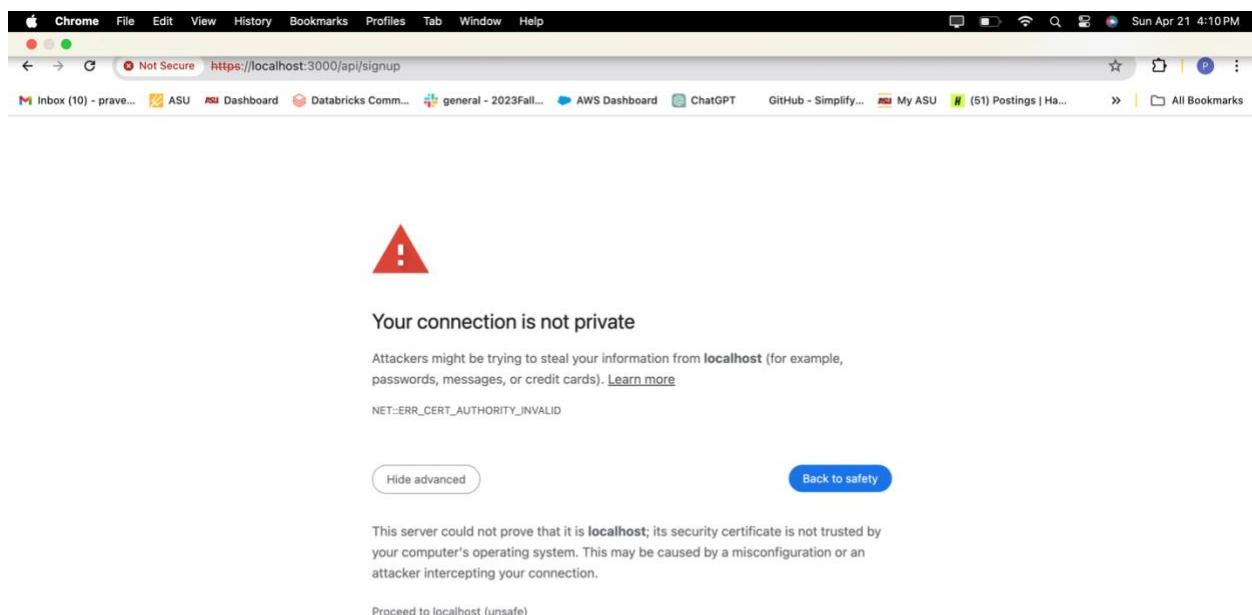
[Running] node "/Users/praveenkumarsiddela/Downloads/M7A1/Server.js"

Server running at <https://localhost:3000>

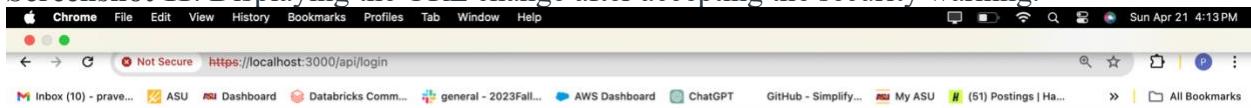
**Screenshot 9:** Showing the browser view when accessing the HTTPS server.



**Screenshot 10:** Showing the browser's security warning.



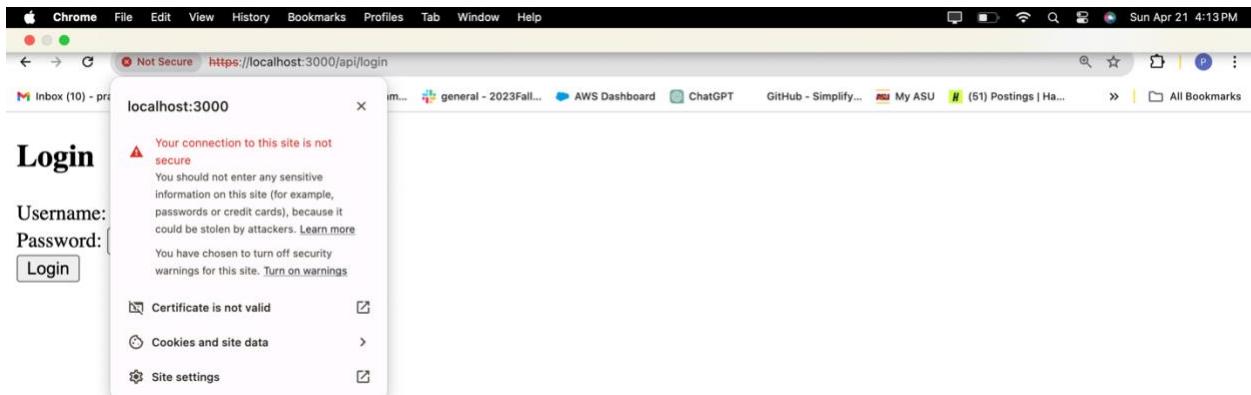
**Screenshot 11:** Displaying the URL change after accepting the security warning.

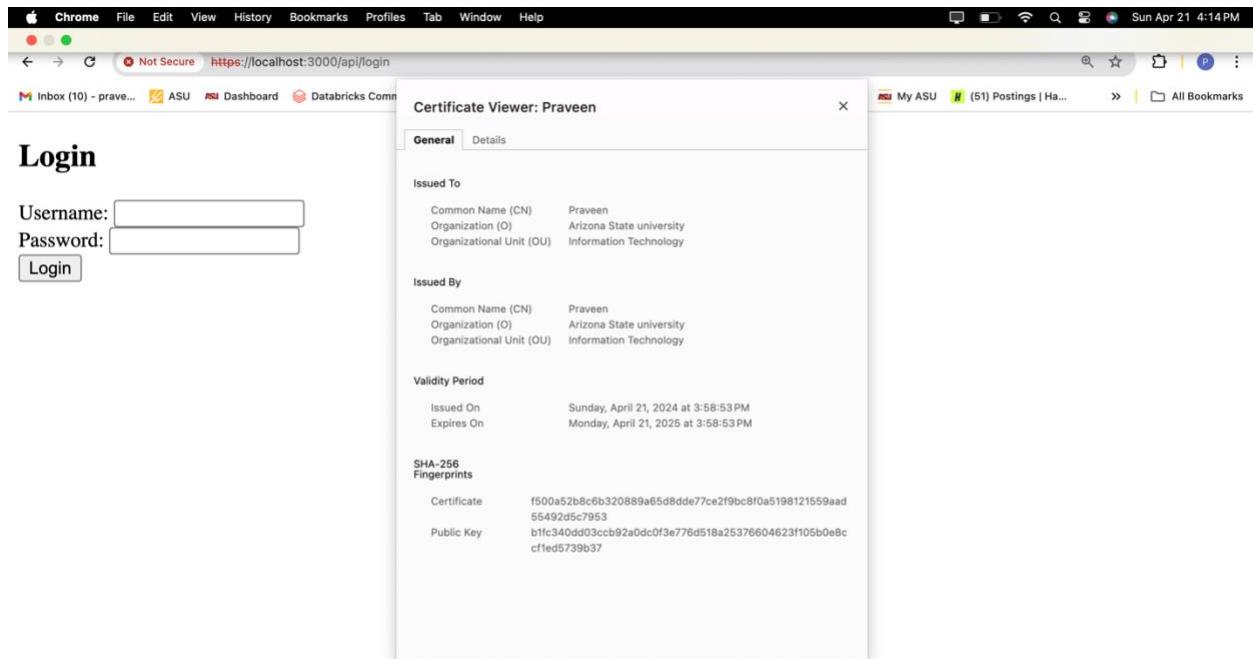


## Login

Username:

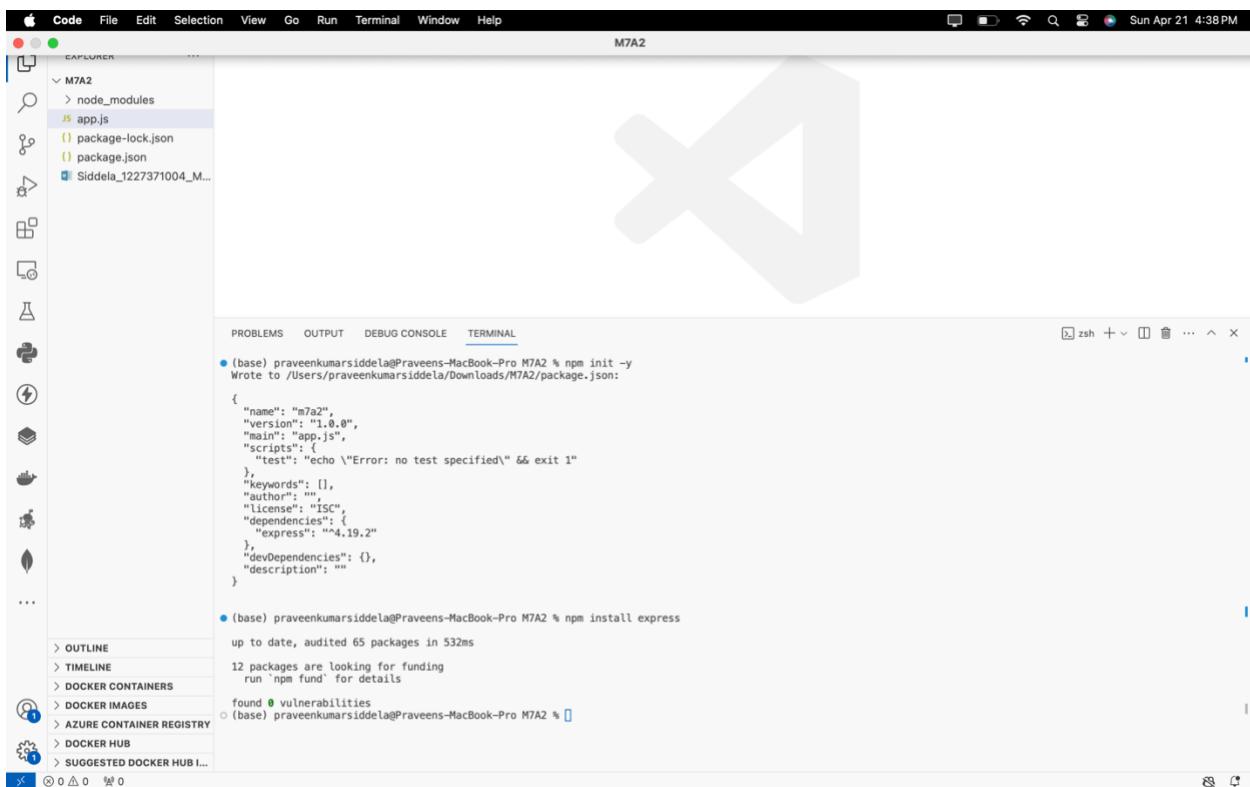
Password:





## Module 7 Activity 2: Implementing HTTP Status Codes and Error Handling in Node.js

Screenshot 1: Of the initialized project structure in your IDE.

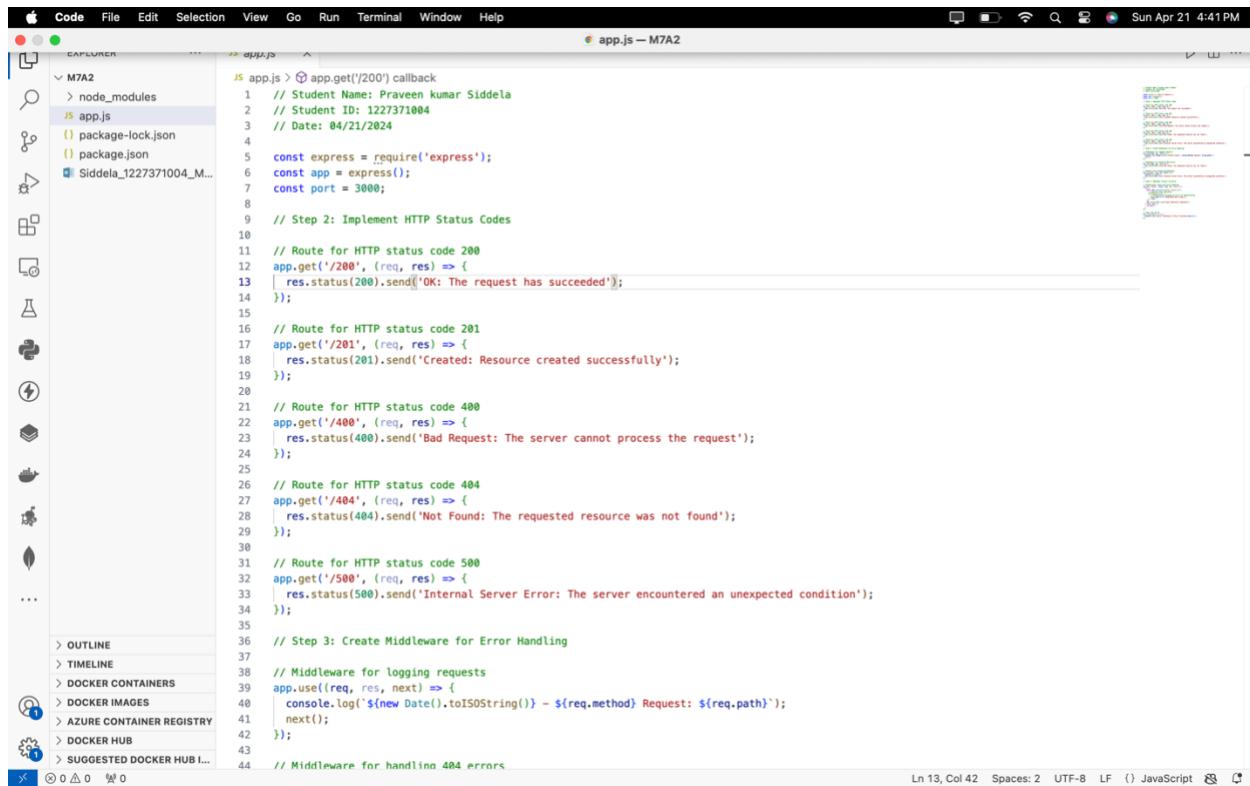


The screenshot shows the Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes Code, File, Edit, Selection, View, Go, Run, Terminal, Window, and Help. The status bar at the top right shows the date and time: Sun Apr 21 4:38 PM. The main area is titled 'M7A2'. The left sidebar (Explorer) shows a project structure with a folder 'M7A2' containing 'node\_modules', 'app.js', 'package-lock.json', and 'package.json'. Below this is a list of Docker-related items: 'OUTLINE', 'TIMELINE', 'DOCKER CONTAINERS', 'DOCKER IMAGES', 'AZURE CONTAINER REGISTRY', 'DOCKER HUB', and 'SUGGESTED DOCKER HUB I...'. The right side of the interface is a terminal window. The terminal tab is selected, showing the command 'npm init -y' being run, which creates a package.json file. The terminal then installs the 'express' package with the command 'npm install express', indicating an audit of 65 packages and 12 packages looking for funding. The terminal concludes with a check for vulnerabilities and a final command run.

```
(base) praveenkumarsiddela@Praveens-MacBook-Pro M7A2 % npm init -y
Wrote to /Users/praveenkumarsiddela/Downloads/M7A2/package.json:
{
  "name": "M7A2",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" & exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "4.19.2"
  },
  "devDependencies": {},
  "description": ""
}

(base) praveenkumarsiddela@Praveens-MacBook-Pro M7A2 % npm install express
up to date, audited 65 packages in 532ms
12 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
(base) praveenkumarsiddela@Praveens-MacBook-Pro M7A2 %
```

**Screenshot 2:** Showing the [app.js](#) file implemented routes for different HTTP status codes.



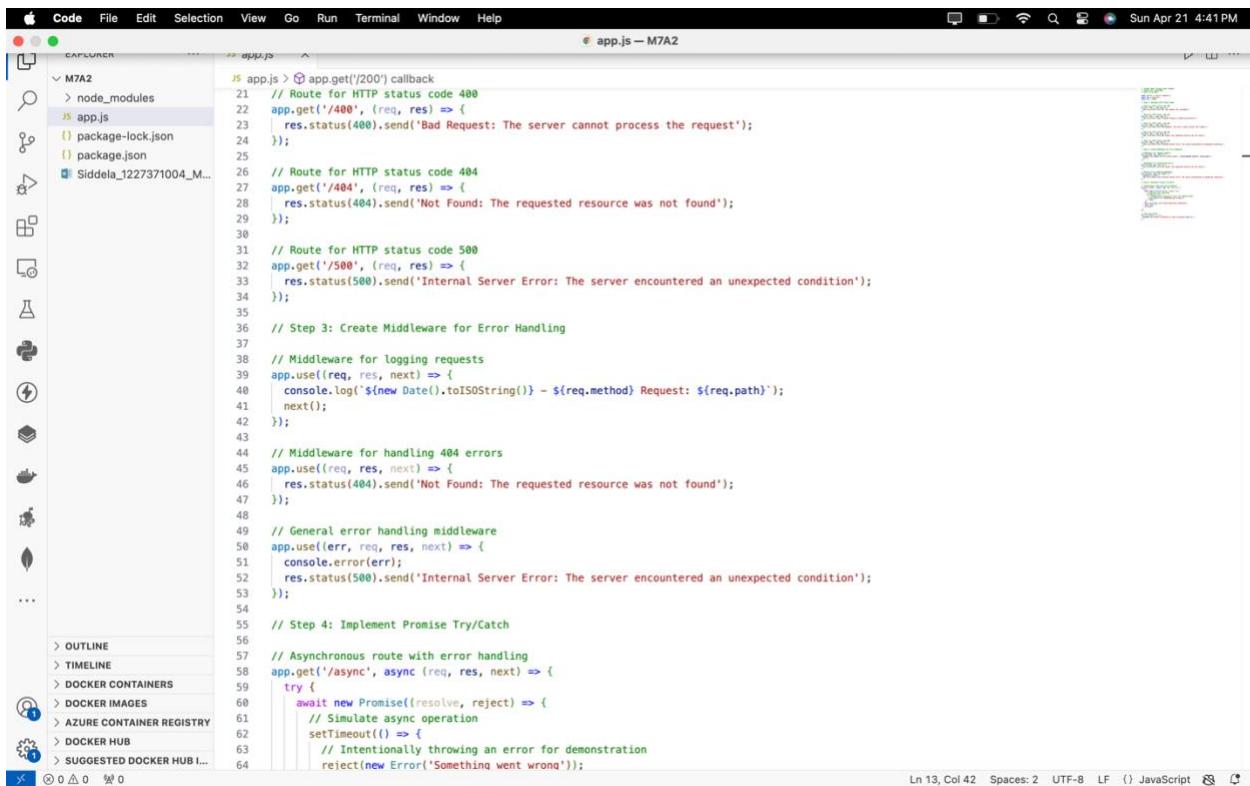
```

 1 // Student Name: Praveen Kumar Siddela
 2 // Student ID: 1227371004
 3 // Date: 04/21/2024
 4
 5 const express = require('express');
 6 const app = express();
 7 const port = 3000;
 8
 9 // Step 2: Implement HTTP Status Codes
10
11 // Route for HTTP status code 200
12 app.get('/200', (req, res) => {
13   res.status(200).send('OK: The request has succeeded');
14 });
15
16 // Route for HTTP status code 201
17 app.get('/201', (req, res) => {
18   res.status(201).send('Created: Resource created successfully');
19 });
20
21 // Route for HTTP status code 400
22 app.get('/400', (req, res) => {
23   res.status(400).send('Bad Request: The server cannot process the request');
24 });
25
26 // Route for HTTP status code 404
27 app.get('/404', (req, res) => {
28   res.status(404).send('Not Found: The requested resource was not found');
29 });
30
31 // Route for HTTP status code 500
32 app.get('/500', (req, res) => {
33   res.status(500).send('Internal Server Error: The server encountered an unexpected condition');
34 });
35
36 // Step 3: Create Middleware for Error Handling
37
38 // Middleware for logging requests
39 app.use((req, res, next) => {
40   console.log(`${new Date().toISOString()} - ${req.method} Request: ${req.path}`);
41   next();
42 });
43
44 // Middleware for handling 404 errors

```

Ln 13, Col 42 Spaces: 2 UTF-8 LF () JavaScript

**Screenshot 3:** Displaying the implemented middleware in `app.js` for request logging and error handling.



The screenshot shows a code editor interface with the following details:

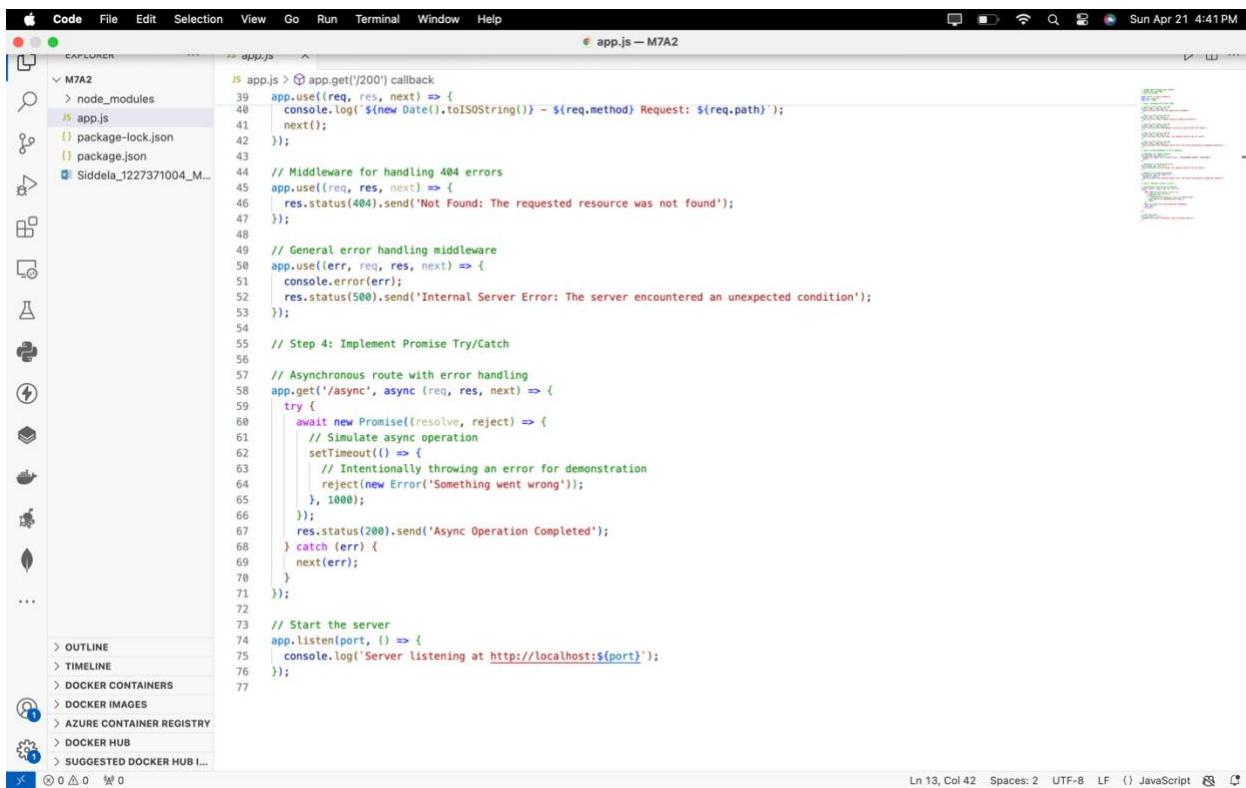
- File Path:** app.js — M7A2
- Code Content:**

```

1 app.get('/200') callback
2 // Route for HTTP status code 400
3 app.get('/400', (req, res) => {
4   res.status(400).send('Bad Request: The server cannot process the request');
5 });
6
7 // Route for HTTP status code 404
8 app.get('/404', (req, res) => {
9   res.status(404).send('Not Found: The requested resource was not found');
10 });
11
12 // Route for HTTP status code 500
13 app.get('/500', (req, res) => {
14   res.status(500).send('Internal Server Error: The server encountered an unexpected condition');
15 });
16
17 // Step 3: Create Middleware for Error Handling
18
19 // Middleware for logging requests
20 app.use((req, res, next) => {
21   console.log(`${new Date().toISOString()} - ${req.method} Request: ${req.path}`);
22   next();
23 });
24
25 // Middleware for handling 404 errors
26 app.use((req, res, next) => {
27   res.status(404).send('Not Found: The requested resource was not found');
28 });
29
30 // General error handling middleware
31 app.use((err, req, res, next) => {
32   console.error(err);
33   res.status(500).send('Internal Server Error: The server encountered an unexpected condition');
34 });
35
36 // Step 4: Implement Promise Try/Catch
37
38 // Asynchronous route with error handling
39 app.get('/async', async (req, res, next) => {
40   try {
41     await new Promise((resolve, reject) => {
42       // Simulate async operation
43       setTimeout(() => {
44         // Intentionally throwing an error for demonstration
45         reject(new Error('Something went wrong'));
46       }, 2000);
47     });
48   } catch (error) {
49     res.status(500).send(`Error: ${error.message}`);
50   }
51 });
52
53
54
55
56
57
58
59
60
61
62
63
64

```
- Editor Status:** Ln 13, Col 42 Spaces: 2 UTF-8 LF (JavaScript)

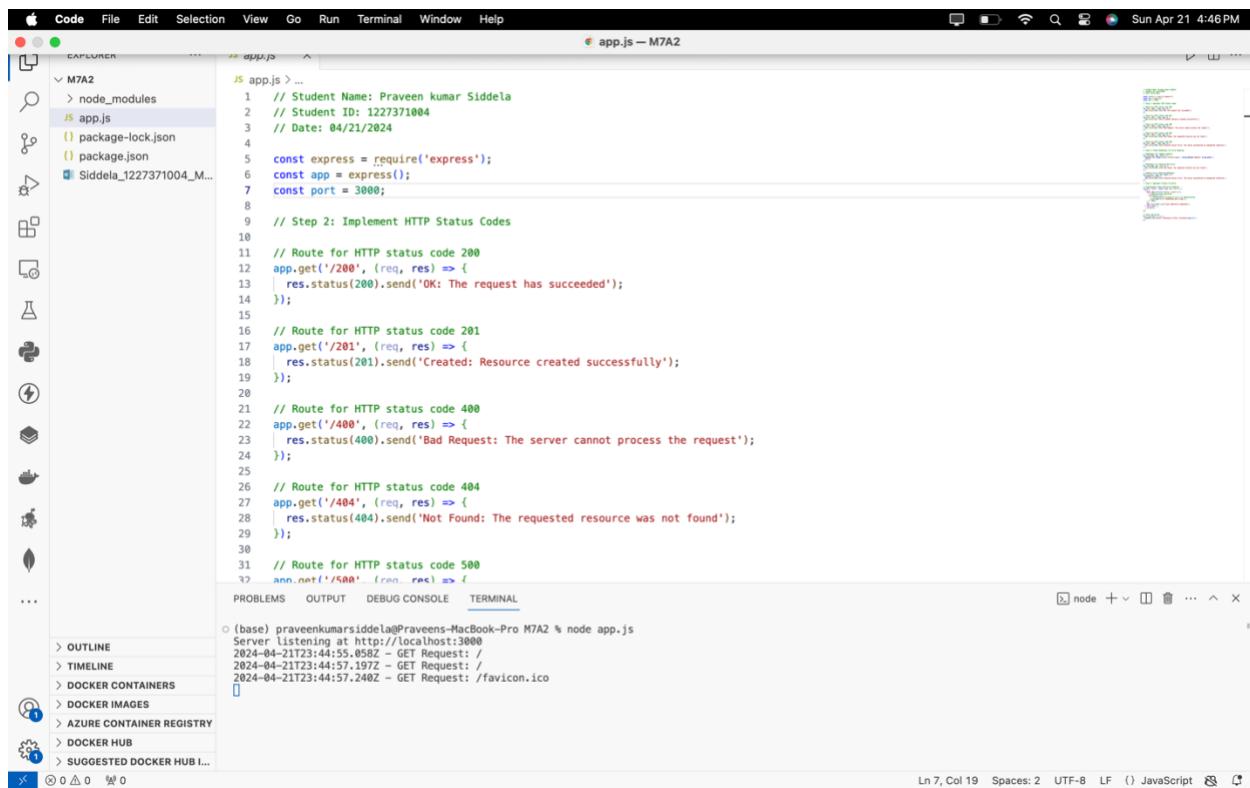
**Screenshot 4:** Of the asynchronous route with try/catch in [app.js](#).



The screenshot shows a code editor interface with a dark theme. The top bar includes the Apple logo, Code, File, Edit, Selection, View, Go, Run, Terminal, Window, and Help menus. The status bar at the bottom right shows the date and time as Sun Apr 21 4:41 PM, and the file path as app.js — M7A2. The bottom status bar also displays line 13, column 42, and other file statistics. The left sidebar contains an Explorer view with a tree structure showing a project named 'M7A2' with subfolders 'node\_modules', 'app.js', 'package-lock.json', 'package.json', and a file 'Siddela\_1227371004\_M...'. Below the Explorer are sections for OUTLINE, TIMELINE, DOCKER CONTAINERS, DOCKER IMAGES, AZURE CONTAINER REGISTRY, DOCKER HUB, and SUGGESTED DOCKER HUB. The main editor area displays the following code:

```
JS app.js > app.get('/200') callback
39 app.use(req, res, next) => {
40   console.log(`New Date(): ${Date().toISOString()} - ${req.method} Request: ${req.path}`);
41   next();
42 };
43
44 // Middleware for handling 404 errors
45 app.use(req, res, next) => {
46   res.status(404).send('Not Found: The requested resource was not found');
47 };
48
49 // General error handling middleware
50 app.use(err, req, res, next) => {
51   console.error(err);
52   res.status(500).send('Internal Server Error: The server encountered an unexpected condition');
53 };
54
55 // Step 4: Implement Promise Try/Catch
56
57 // Asynchronous route with error handling
58 app.get('/async', async (req, res, next) => {
59   try {
60     await new Promise((resolve, reject) => {
61       // Simulate async operation
62       setTimeout(() => {
63         // Intentionally throwing an error for demonstration
64         reject(new Error('Something went wrong'));
65       }, 1000);
66     });
67     res.status(200).send('Async Operation Completed');
68   } catch (err) {
69     next(err);
70   }
71 });
72
73 // Start the server
74 app.listen(port, () => {
75   console.log(`Server listening at http://localhost:${port}`);
76 });
77
```

**Screenshot 5:** Showing the running server in the terminal.



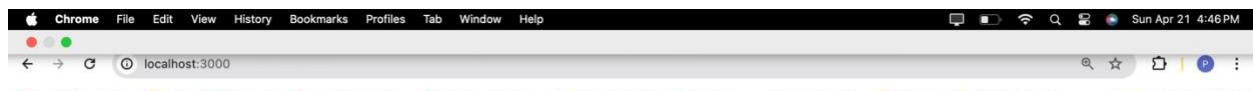
The screenshot shows a Code editor interface with the following details:

- File Explorer:** Shows a project structure for "M7A2" with files: node\_modules, app.js, package-lock.json, package.json, and a partially visible file "Siddela\_1227371004\_M...".
- Code Editor:** The "app.js" file is open, displaying the following code:
 

```

1 // Student Name: Praveen kumar Siddela
2 // Student ID: 1227371004
3 // Date: 04/21/2024
4
5 const express = require('express');
6 const app = express();
7 const port = 3000;
8
9 // Step 2: Implement HTTP Status Codes
10
11 // Route for HTTP status code 200
12 app.get('/200', (req, res) => {
13   res.status(200).send('OK: The request has succeeded');
14 });
15
16 // Route for HTTP status code 201
17 app.get('/201', (req, res) => {
18   res.status(201).send('Created: Resource created successfully');
19 });
20
21 // Route for HTTP status code 400
22 app.get('/400', (req, res) => {
23   res.status(400).send('Bad Request: The server cannot process the request');
24 });
25
26 // Route for HTTP status code 404
27 app.get('/404', (req, res) => {
28   res.status(404).send('Not Found: The requested resource was not found');
29 });
30
31 // Route for HTTP status code 500
32 app.get('/500', (req, res) => {
33   res.status(500).send('Internal Server Error: The server has encountered an unexpected error');
34 });
      
```
- Terminal:** The terminal window shows the command "node app.js" being run and the output of the server listening on port 3000. It also shows three GET requests from the browser to the server's IP address and port 3000.

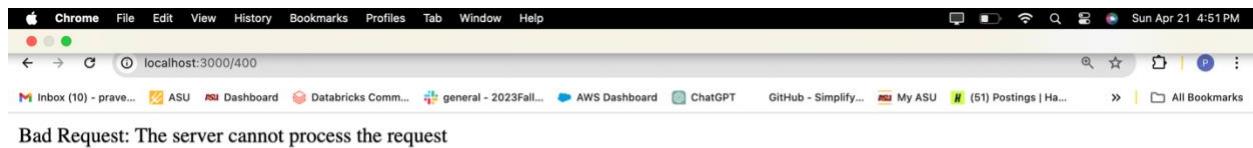
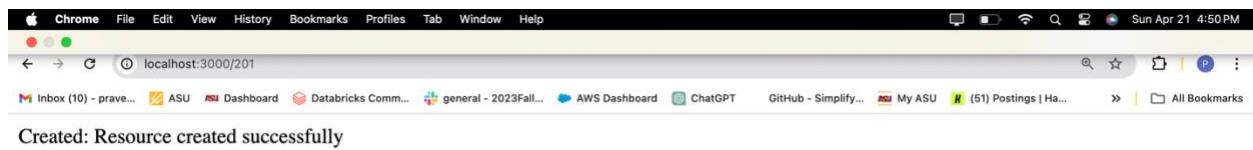
**Screenshot 6:** Displaying responses from the server for different routes in Postman or a web browser



Not Found: The requested resource was not found



OK: The request has succeeded





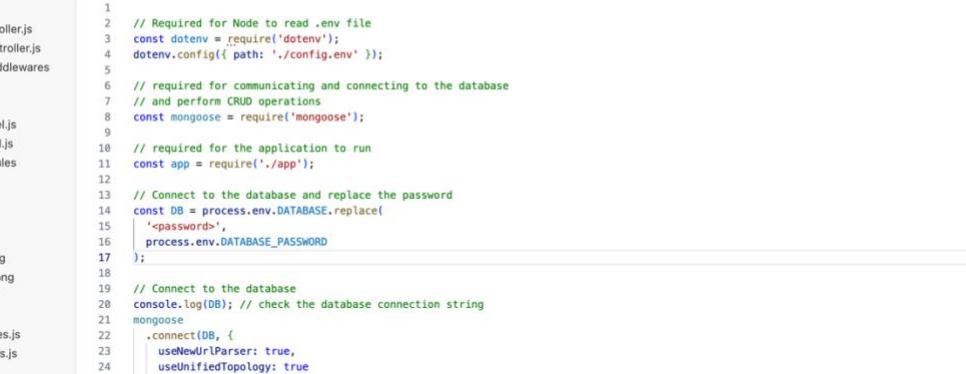
Not Found: The requested resource was not found



Internal Server Error: The server encountered an unexpected condition

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

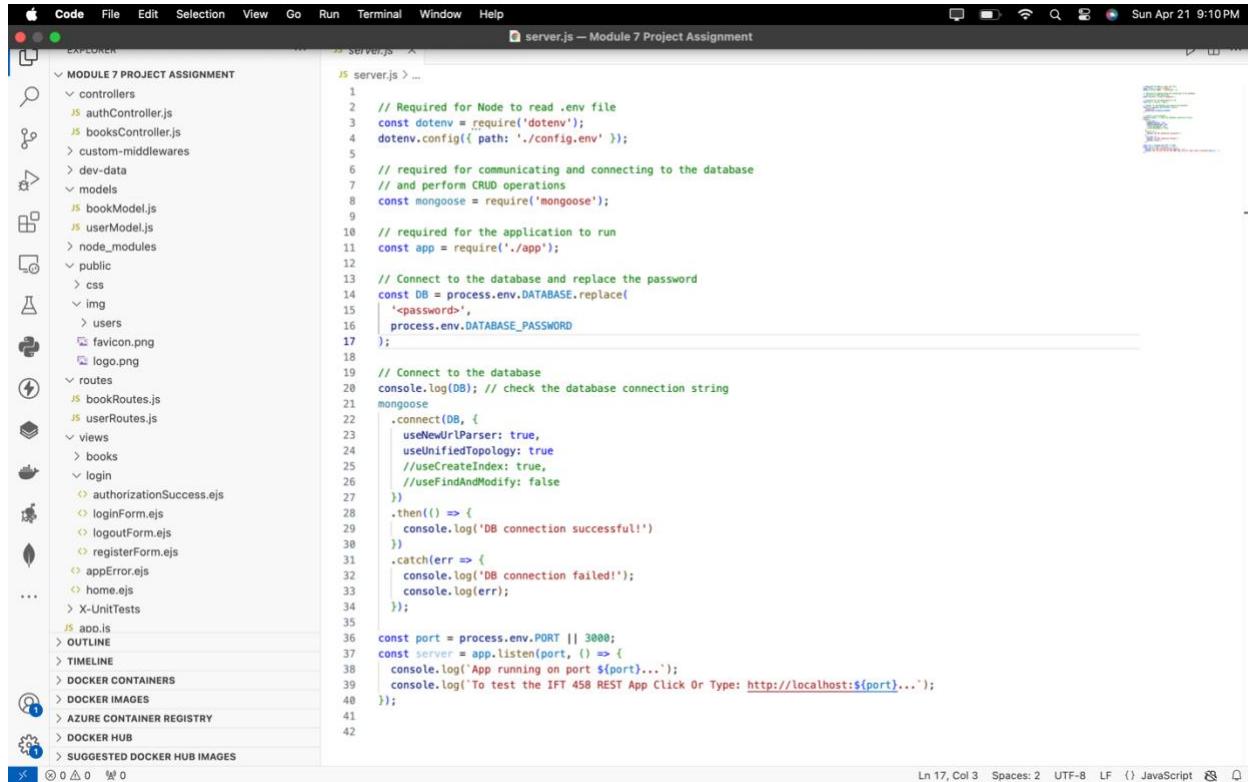
**Screenshot 1:** Of the terminal showing successful installation of dependencies.



The screenshot shows a code editor with a sidebar and a main workspace. The sidebar on the left contains project navigation, file lists for controllers, models, routes, and views, and sections for X-UnitTests, OUTLINE, TIMELINE, DOCKER CONTAINERS, DOCKER IMAGES, AZURE CONTAINER REGISTRY, DOCKER HUB, and SUGGESTED DOCKER HUB. The main workspace shows a file named 'server.js' with the following content:

```
JS server.js > ...
1 // Required for Node to read .env file
2 const dotenv = require('dotenv');
3 dotenv.config({ path: './config.env' });
4
5
6 // required for communicating and connecting to the database
7 // and perform CRUD operations
8 const mongoose = require('mongoose');
9
10 // required for the application to run
11 const app = require('./app');
12
13 // Connect to the database and replace the password
14 const DB = process.env.DATABASE.replace(
15   '<password>',
16   process.env.DATABASE_PASSWORD
17 );
18
19 // Connect to the database
20 console.log(DB); // check the database connection string
21 mongoose
22   .connect(DB, {
23     useNewUrlParser: true,
24     useUnifiedTopology: true,
25     //useCreateIndex: true,
26   });
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2057
2058
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2117
2118
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2127
2128
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2157
2158
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2167
2168
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2177
2178
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2217
2218
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2227
2228
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2237
2238
2238
2239
2
```

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



The screenshot shows a code editor interface with the following details:

- File Path:** server.js – Module 7 Project Assignment
- Code Content (server.js):**

```

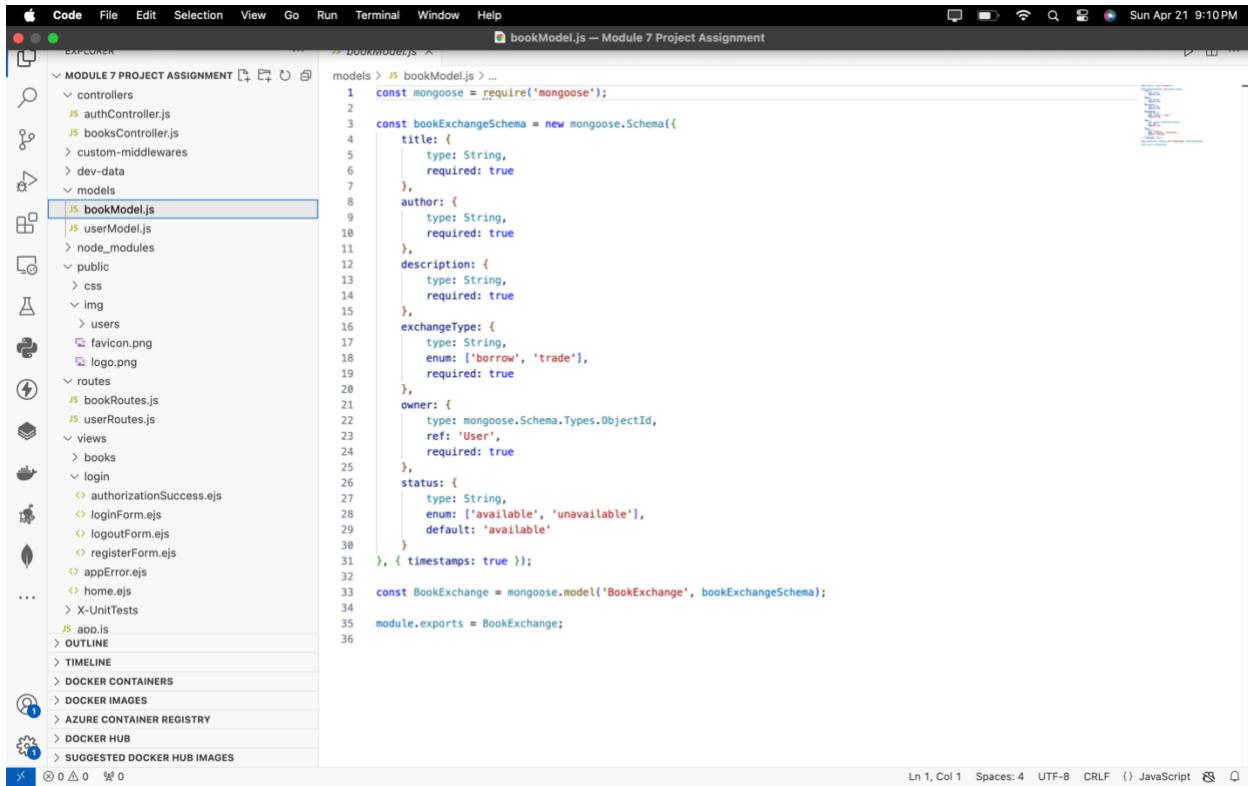
1  // Required for Node to read .env file
2  const dotenv = require('dotenv');
3  dotenv.config({ path: './config.env' });
4
5  // required for communicating and connecting to the database
6  // and perform CRUD operations
7  const mongoose = require('mongoose');
8
9  // required for the application to run
10 const app = require('./app');
11
12
13 // Connect to the database and replace the password
14 const DB = process.env.DATABASE.replace(
15   '<password>',
16   process.env.DATABASE_PASSWORD
17 );
18
19 // Connect to the database
20 console.log(DB); // check the database connection string
21 mongoose
22   .connect(DB, {
23     useNewUrlParser: true,
24     useUnifiedTopology: true,
25     //useCreateIndex: true,
26     //useFindAndModify: false
27   })
28   .then(() => {
29     console.log('DB connection successful!')
30   })
31   .catch(err => {
32     console.log('DB connection failed!');
33     console.log(err);
34   });
35
36 const port = process.env.PORT || 3000;
37 const server = app.listen(port, () => {
38   console.log(`App running on port ${port}...`);
39   console.log(`To test the IFT 458 REST App Click Or Type: http://localhost:\${port}...`);
40 });
41
42

```

- Project Structure (Explorer):**
  - MODULE 7 PROJECT ASSIGNMENT
    - controllers
      - authController.js
      - booksController.js
    - custom\_middlewares
    - dev-data
    - models
      - bookModel.js
      - userModel.js
    - node\_modules
    - public
      - css
      - img
        - users
        - favicon.png
        - logo.png
    - routes
      - bookRoutes.js
      - userRoutes.js
    - views
      - books
      - login
        - authorizationSuccess.ejs
        - loginForm.ejs
        - logoutForm.ejs
        - registerForm.ejs
      - appError.ejs
      - home.ejs
    - X-UnitTests
    - app.js
    - OUTLINE
    - OUTLINE
    - DOCKER CONTAINERS
    - DOCKER IMAGES
    - AZURE CONTAINER REGISTRY
    - DOCKER HUB
    - SUGGESTED DOCKER HUB IMAGES

Ln 17, Col 3 Spaces: 2 UTF-8 LF {} JavaScript

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



```

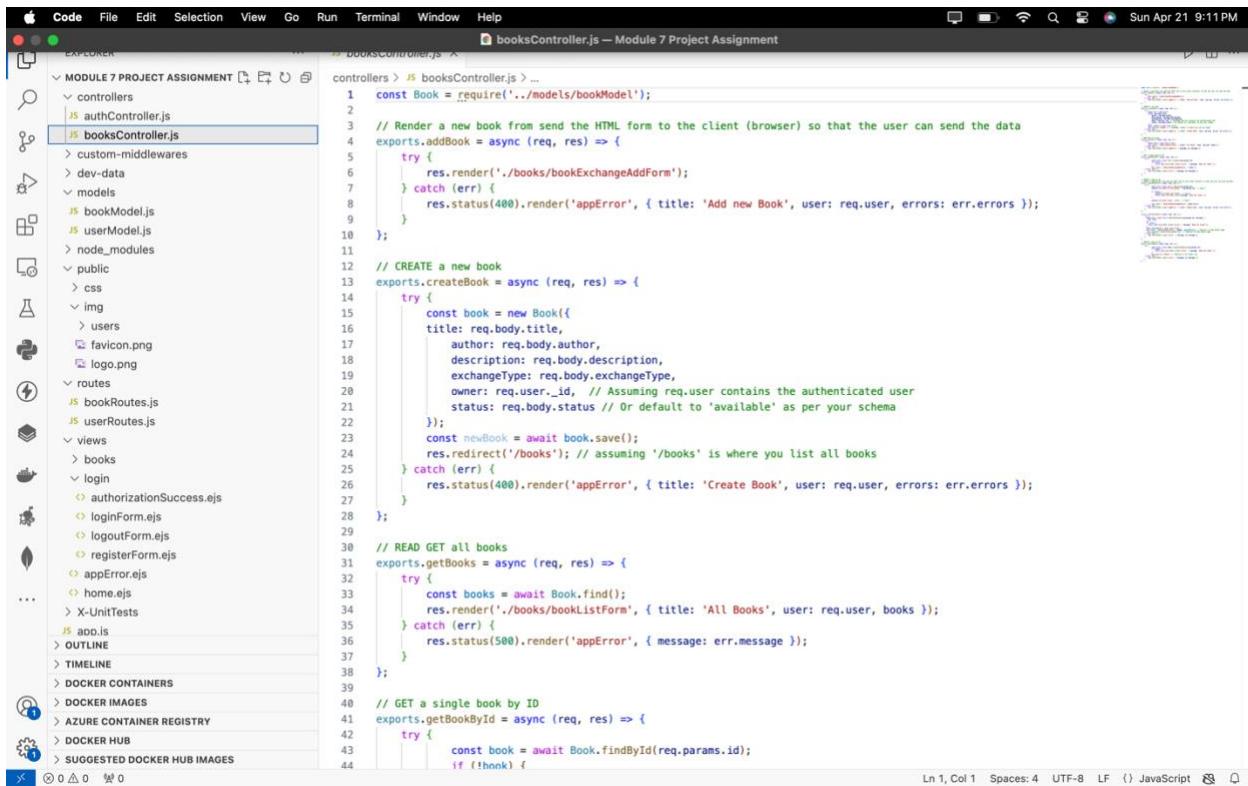
1  const mongoose = require('mongoose');
2
3  const bookExchangeSchema = new mongoose.Schema({
4    title: {
5      type: String,
6      required: true
7    },
8    author: {
9      type: String,
10     required: true
11    },
12    description: {
13      type: String,
14      required: true
15    },
16    exchangeType: {
17      type: String,
18      enum: ['borrow', 'trade'],
19      required: true
20    },
21    owner: {
22      type: mongoose.Schema.Types.ObjectId,
23      ref: 'User',
24      required: true
25    },
26    status: {
27      type: String,
28      enum: ['available', 'unavailable'],
29      default: 'available'
30    }
31  }, { timestamps: true });
32
33  const BookExchange = mongoose.model('BookExchange', bookExchangeSchema);
34
35  module.exports = BookExchange;

```

The **bookModel.js** file defines the book data structure for the application using Mongoose. It specifies properties like title, author, description, exchangeType, owner, and status, with precise data types and constraints. Enums ensure consistency for exchangeType and status. The owner field references the 'User' model. Default values and timestamps are included. The schema becomes a Mongoose model named 'BookExchange' for easy MongoDB interaction, exported for application-wide use.

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

The **bookController.js** file handles book-related actions in the app. It includes functions for adding, retrieving, updating, and deleting books. Error handling ensures appropriate status codes and messages. Functions like addBook and updateBookForm handle user input, while createBook, getBooks, getBookById, updateBookById, and deleteBook interact with the database. Status codes like 400, 404, and 500 are used for various scenarios, ensuring smooth operation of book-related functionalities.

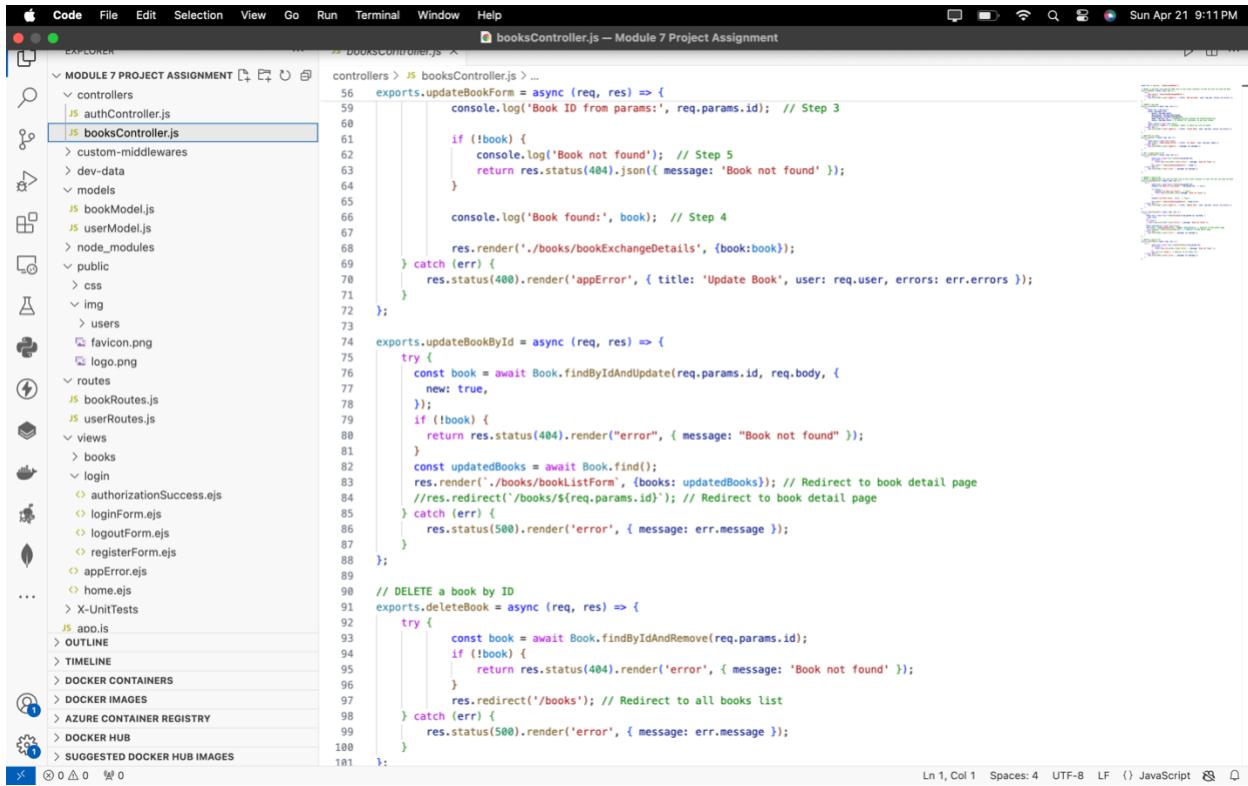


```

 1  const Book = require('../models/bookModel');
 2
 3  // Render a new book from 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, // Assuming req.user contains the authenticated user
21        status: req.body.status // Or default to 'available' as per your schema
22      });
23      const newBook = await book.save();
24      res.redirect('/books'); // assuming '/books' is where you list all 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
31  exports.getBooks = async (req, res) => {
32    try {
33      const books = await Book.find();
34      res.render('../books/bookListForm', { title: 'All Books', user: req.user, books });
35    } catch (err) {
36      res.status(500).render('appError', { message: err.message });
37    }
38  };
39
40  // GET a single book by ID
41  exports.getBookById = async (req, res) => {
42    try {
43      const book = await Book.findById(req.params.id);
44      if (!book) {

```

Ln 1, Col 1 Spaces: 4 UTF-8 LF () JavaScript

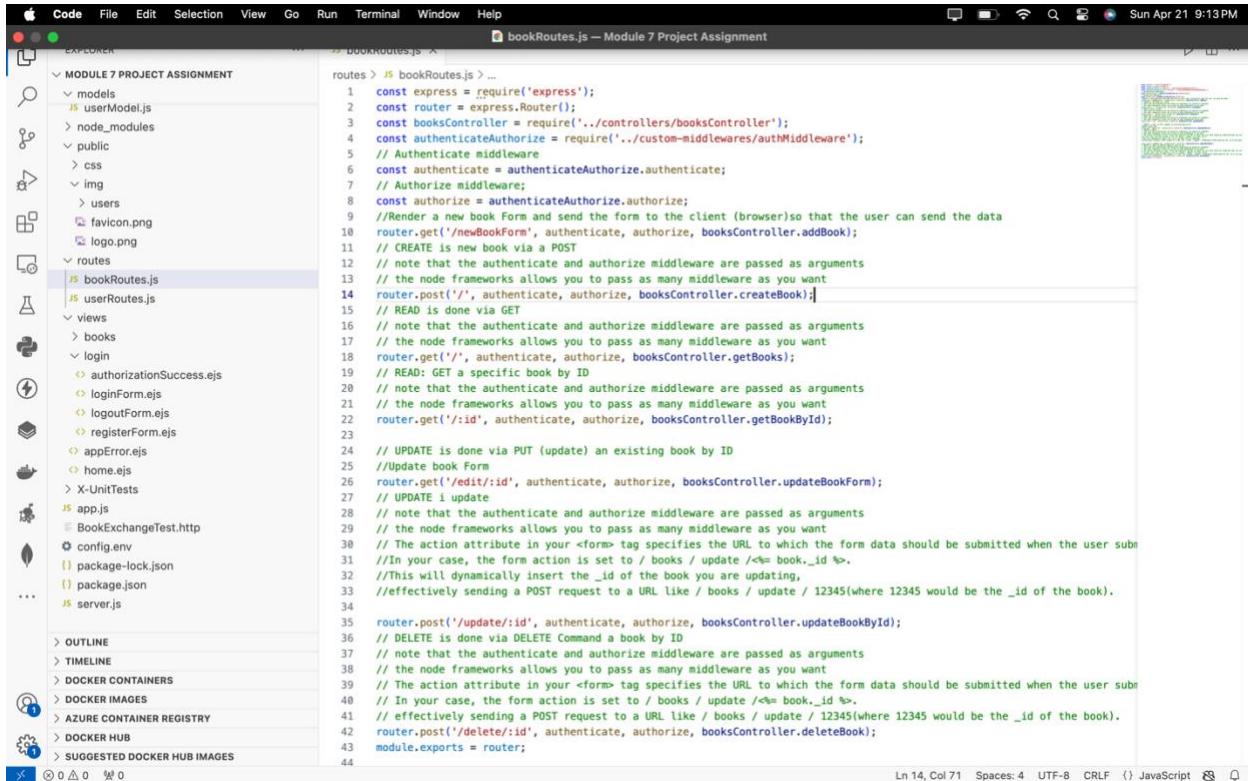


```

Code File Edit Selection View Go Run Terminal Window Help
booksController.js — Module 7 Project Assignment
EXPLORER
MODULE 7 PROJECT ASSIGNMENT
controllers
  authController.js
  booksController.js
  custom-middlewares
  dev-data
models
  bookModel.js
  userModel.js
node_modules
public
  css
  img
  users
    favicon.png
    logo.png
routes
  bookRoutes.js
  userRoutes.js
views
  books
  login
    authorizationSuccess.ejs
    loginForm.ejs
    logoutForm.ejs
    registerForm.ejs
    appError.ejs
    home.ejs
  X-UnitTests
  app.js
  BookExchangeTest.http
  config.env
  package-lock.json
  package.json
  server.js
  OUTLINE
  TIMELINE
  DOCKER CONTAINERS
  DOCKER IMAGES
  AZURE CONTAINER REGISTRY
  DOCKER HUB
  SUGGESTED DOCKER HUB IMAGES
booksController.js
56  exports.updateBookForm = async (req, res) => {
57    console.log('Book ID from params:', req.params.id); // Step 3
58
59    if (!book) {
60      console.log('Book not found'); // Step 5
61      return res.status(404).json({ message: 'Book not found' });
62    }
63
64    console.log('Book found:', book); // Step 4
65
66    res.render('../books/bookExchangeDetails', {book:book});
67
68  } catch (err) {
69    res.status(400).render('appError', { title: 'Update Book', user: req.user, errors: err.errors });
70  }
71
72  };
73
74  exports.updateBookById = async (req, res) => {
75    try {
76      const book = await Book.findByIdAndUpdate(req.params.id, req.body, {
77        new: true,
78      });
79      if (!book) {
80        return res.status(404).render("error", { message: "Book not found" });
81      }
82      const updatedBooks = await Book.find();
83      res.render('../books/bookListForm', {books: updatedBooks}); // Redirect to book detail page
84      //res.redirect('/books/${req.params.id}'); // Redirect to book detail page
85    } catch (err) {
86      res.status(500).render('error', { message: err.message });
87    }
88  };
89
90  // DELETE a book by ID
91  exports.deleteBook = async (req, res) => {
92    try {
93      const book = await Book.findByIdAndRemove(req.params.id);
94      if (!book) {
95        return res.status(404).render('error', { message: 'Book not found' });
96      }
97      res.redirect('/books'); // Redirect to all books list
98    } catch (err) {
99      res.status(500).render('error', { message: err.message });
100    }
101  };

```

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



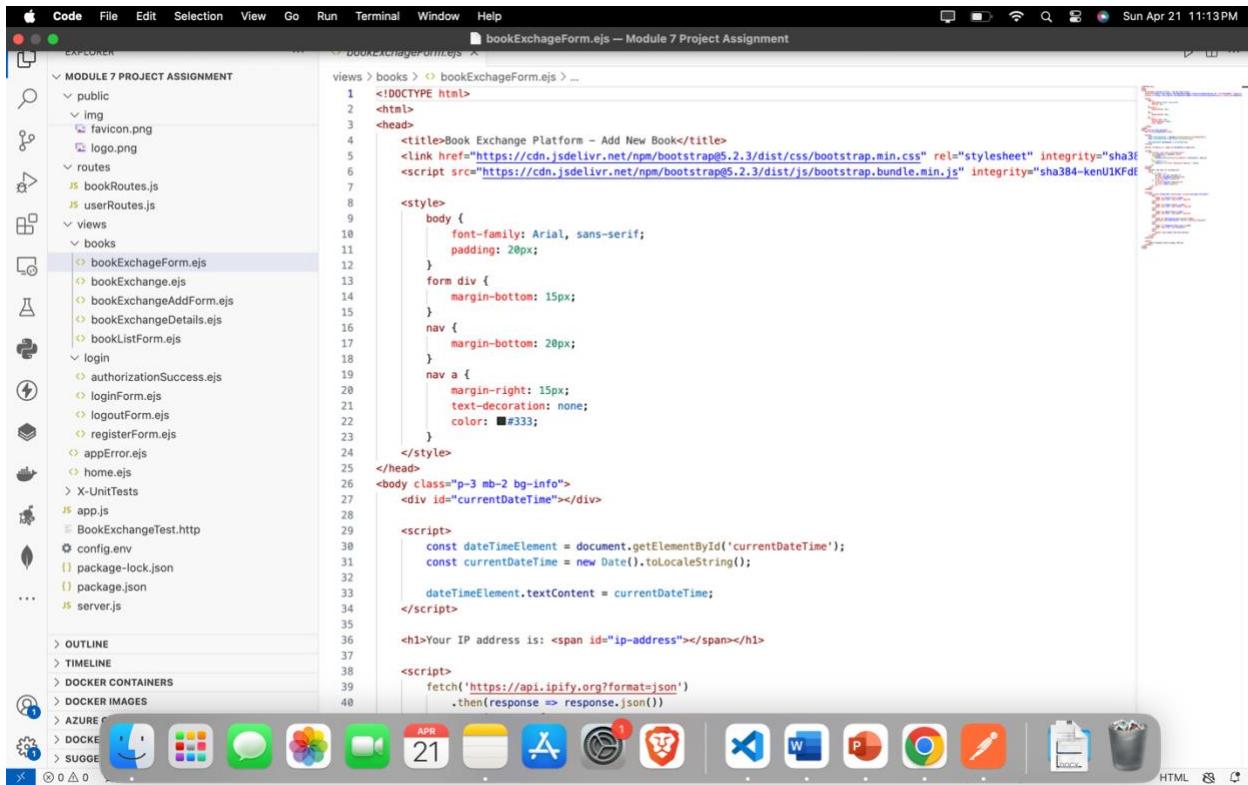
```

Code File Edit Selection View Go Run Terminal Window Help
bookRoutes.js — Module 7 Project Assignment
EXPLORER
MODULE 7 PROJECT ASSIGNMENT
models
  userModel.js
node_modules
public
  css
  img
  users
    favicon.png
    logo.png
routes
  bookRoutes.js
  userRoutes.js
views
  books
  login
    authorizationSuccess.ejs
    loginForm.ejs
    logoutForm.ejs
    registerForm.ejs
    appError.ejs
    home.ejs
  X-UnitTests
  app.js
  BookExchangeTest.http
  config.env
  package-lock.json
  package.json
  server.js
  OUTLINE
  TIMELINE
  DOCKER CONTAINERS
  DOCKER IMAGES
  AZURE CONTAINER REGISTRY
  DOCKER HUB
  SUGGESTED DOCKER HUB IMAGES
bookRoutes.js
1  const express = require('express');
2  const router = express.Router();
3  const booksController = require('../controllers/booksController');
4  const authenticateAuthorize = require('../custom-middlewares/authMiddleware');
5  // Authenticate middleware
6  const authenticate = authenticateAuthorize.authenticate;
7  // Authorize middleware;
8  const authorize = authenticateAuthorize.authorize;
9  //Render a new book Form and send the form to the client (browser)so that the user can send the data
10 router.get('/newBookForm', authenticate, authorize, booksController.addBook);
11 // CREATE is new book via a POST
12 // note that the authenticate and authorize middleware are passed as arguments
13 // the node frameworks allows you to pass as many middleware as you want
14 router.post('/', authenticate, authorize, booksController.createBook);
15 // READ is done via GET
16 // note that the authenticate and authorize middleware are passed as arguments
17 // the node frameworks allows you to pass as many middleware as you want
18 router.get('/', authenticate, authorize, booksController.getBooks);
19 // READ: GET a specific book by ID
20 // note that the authenticate and authorize middleware are passed as arguments
21 // the node frameworks allows you to pass as many middleware as you want
22 router.get('/:id', authenticate, authorize, booksController.getBookById);
23
24 // UPDATE is done via PUT (update) an existing book by ID
25 //Update book Form
26 router.get('/edit/:id', authenticate, authorize, booksController.updateBookForm);
27 // UPDATE i update
28 // note that the authenticate and authorize middleware are passed as arguments
29 // the node frameworks allows you to pass as many middleware as you want
30 // The action attribute in your <form> tag specifies the URL to which the form data should be submitted when the user submits the form.
31 //In your case, the form action is set to / books / update /<%= book._id %>.
32 //This will dynamically insert the _id of the book you are updating,
33 //effectively sending a POST request to a URL like / books / update / 12345(where 12345 would be the _id of the book).
34
35 router.post('/update/:id', authenticate, authorize, booksController.updateBookById);
36 // DELETE is done via DELETE Command a book by ID
37 // note that the authenticate and authorize middleware are passed as arguments
38 // the node frameworks allows you to pass as many middleware as you want
39 // The action attribute in your <form> tag specifies the URL to which the form data should be submitted when the user submits the form.
40 // In your case, the form action is set to / books / update /<%= book._id %>.
41 // effectively sending a POST request to a URL like / books / update / 12345(where 12345 would be the _id of the book).
42 router.post('/delete/:id', authenticate, authorize, booksController.deleteBook);
43 module.exports = router;

```

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

The **bookExchangeForm.ejs** page is a web form for adding new books to a book exchange platform. It contains standard HTML structure with metadata and external resources like Bootstrap CSS and JavaScript files. The body section dynamically displays date, time, and user IP. When logged in, it shows a welcome message and links to dashboard and logout; otherwise, it provides signup/login buttons. The main section houses a form for entering book details and submitting them to **/books/add**. The footer includes copyright information. Overall, it serves as a user-friendly interface for adding books to the platform.



```

 1 <!DOCTYPE html>
 2 <html>
 3   <head>
 4     <title>Book Exchange Platform - Add New Book</title>
 5     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-...
 6     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-...
 7
 8   <style>
 9     body {
10       font-family: Arial, sans-serif;
11       padding: 20px;
12     }
13     form div {
14       margin-bottom: 15px;
15     }
16     nav {
17       margin-bottom: 20px;
18     }
19     nav a {
20       margin-right: 15px;
21       text-decoration: none;
22       color: #333;
23     }
24   </style>
25 </head>
26 <body class="p-3 mb-2 bg-info">
27   <div id="currentDateTime"></div>
28
29 <script>
30   const dateTimeElement = document.getElementById('currentDateTime');
31   const currentDateTime = new Date().toLocaleString();
32
33   dateTimeElement.textContent = currentDateTime;
34 </script>
35
36 <h1>Your IP address is: <span id="ip-address"></span></h1>
37
38 <script>
39   fetch('https://api.ipify.org?format=json')
40     .then(response => response.json())

```

Code File Edit Selection View Go Run Terminal Window Help

bookExchange.ejs — Module 7 Project Assignment

```

<!DOCTYPE html>
<html>
<head>
  <title>Book Exchange Platform - Available Books</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-...
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-...
<style>
  body {
    font-family: Arial, sans-serif;
    padding: 20px;
  }
  .book {
    border: 1px solid #ccc;
    padding: 10px;
    margin: 10px 0;
  }
  nav {
    margin-bottom: 20px;
  }
  nav a {
    margin-right: 15px;
    text-decoration: none;
    color: #333;
  }
</style>
</head>
<body class="p-3 mb-2 bg-info">
  <div id="currentDateTime"></div>
  <script>
    const dateTimeElement = document.getElementById('currentDateTime');
    const currentDateTime = new Date().toLocaleString();
    dateTimeElement.textContent = currentDateTime;
  </script>
  <h1>Your IP address is: <span id="ip-address"></span></h1>
  <script>
    ...
  </script>
</body>

```

views > books > bookExchange.ejs

bookExchangeForm.ejs

bookExchange.ejs

bookExchangeAddForm.ejs

bookExchangeDetails.ejs

bookListForm.ejs

login

authorizationSuccess.ejs

loginForm.ejs

logoutForm.ejs

registerForm.ejs

appError.ejs

home.ejs

X-UnitTests

app.js

BookExchangeTest.http

config.env

package-lock.json

package.json

server.js

OUTLINE

TIMELINE

DOCKER CONTAINERS

DOCKER IMAGES

AZURE CONTAINERS

DOCKERSPACES

SUGGESTIONS

HTML

Code File Edit Selection View Go Run Terminal Window Help

home.ejs — Module 7 Project Assignment

```

<!DOCTYPE html>
<html>
<head>
  <title>Book Exchange Platform</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-...
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-...
  <style>
    /* Add some basic styles for readability */
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
    }
    .book {
      border: 1px solid #ccc;
      padding: 10px;
      margin: 10px 0;
    }
    nav {
      margin-bottom: 20px;
    }
    nav a {
      margin-right: 15px;
      text-decoration: none;
      color: #333;
    }
  </style>
</head>
<body class="p-3 mb-2 bg-info">
  <div>
    <label>Batch of IFT 458 Fall 2023 </label>
    
  </div>
</body>

```

views > home.ejs

home.ejs

bookExchangeForm.ejs

bookExchange.ejs

bookExchangeAddForm.ejs

bookExchangeDetails.ejs

bookListForm.ejs

login

authorizationSuccess.ejs

loginForm.ejs

logoutForm.ejs

registerForm.ejs

appError.ejs

X-UnitTests

app.js

BookExchangeTest.http

config.env

package-lock.json

package.json

server.js

OUTLINE

TIMELINE

DOCKER CONTAINERS

DOCKER IMAGES

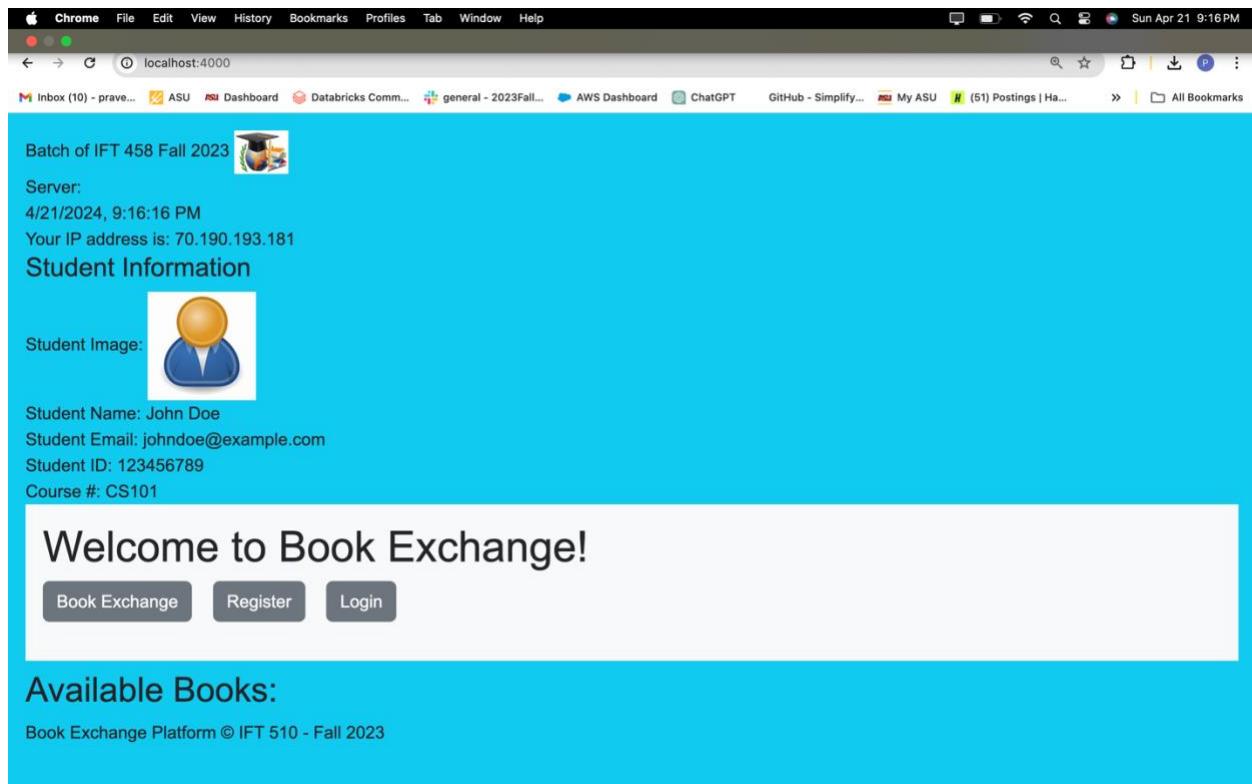
AZURE CONTAINERS

DOCKERSPACES

SUGGESTIONS

HTML

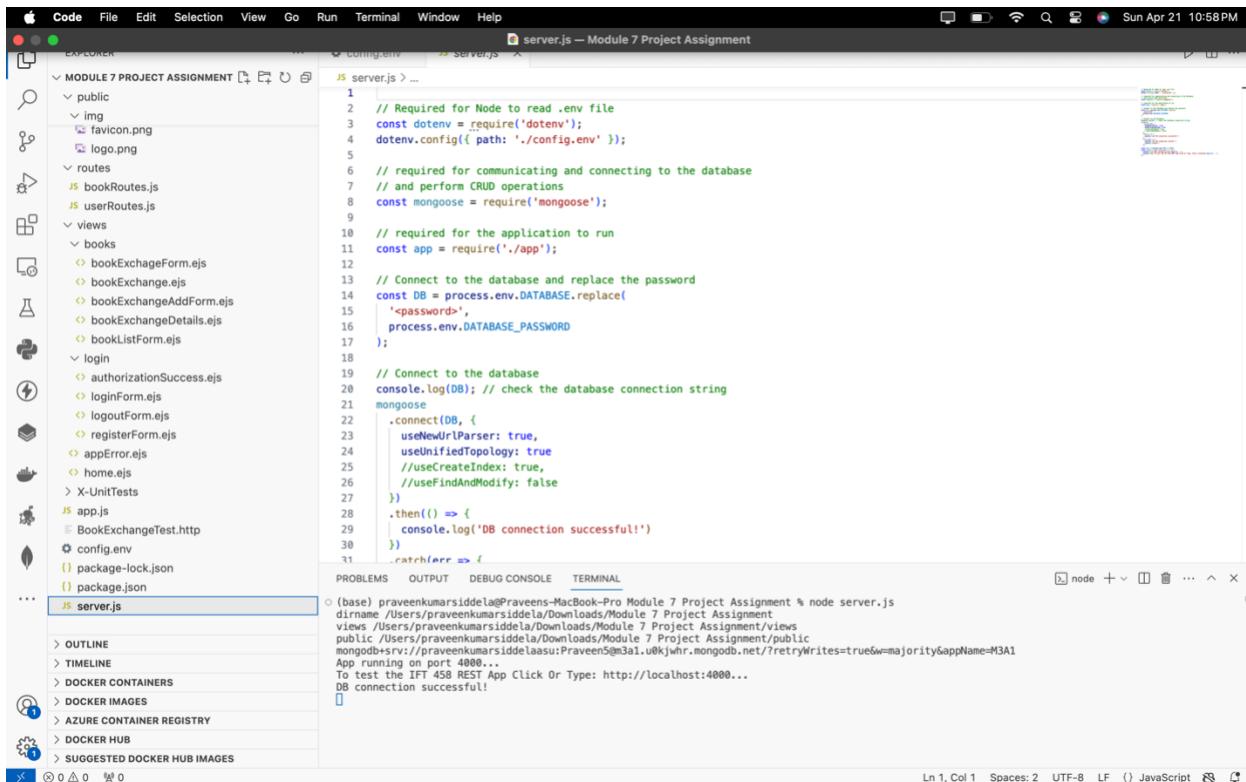
**Screenshot 7:** Of the home page as displayed in the browser.



The screenshot shows a web browser window with the following details:

- Header:** Chrome, File, Edit, View, History, Bookmarks, Profiles, Tab, Window, Help. The address bar shows "localhost:4000".
- Page Content:**
  - Header: "Batch of IFT 458 Fall 2023" with a graduation cap icon.
  - Text: "Server: 4/21/2024, 9:16:16 PM", "Your IP address is: 70.190.193.181".
  - Section: "Student Information" with a placeholder "Student Image" showing a blue graduation cap icon.
  - Text: "Student Name: John Doe", "Student Email: johndoe@example.com", "Student ID: 123456789", "Course #: CS101".
  - Section: "Welcome to Book Exchange!" with buttons for "Book Exchange", "Register", and "Login".
  - Section: "Available Books:" with the text "Book Exchange Platform © IFT 510 - Fall 2023".

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



```

1 // Required for Node to read .env file
2 const dotenv = require('dotenv');
3 dotenv.config({ path: './config.env' });
4
5 // required for communicating and connecting to the database
6 // and perform CRUD operations
7 const mongoose = require('mongoose');
8
9 // required for the application to run
10 const app = require('./app');
11
12 // Connect to the database and replace the password
13 const DB = process.env.DATABASE.replace(
14   '<password>',
15   process.env.DATABASE_PASSWORD
16 );
17
18 // Connect to the database
19 console.log(DB); // check the database connection string
20 mongoose
21   .connect(DB, {
22     useNewUrlParser: true,
23     useUnifiedTopology: true
24     //useCreateIndex: true,
25     //useFindAndModify: false
26   })
27   .then(() => {
28     console.log('DB connection successful!')
29   })
30   .catch(err => {
31     console.log(err)
32   })

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

(base) praveenkumarsiddela@Praveens-MacBook-Pro Module 7 Project Assignment % node server.js

dirname /Users/praveenkumarsiddela/Downloads/Module 7 Project Assignment

views /Users/praveenkumarsiddela/Downloads/Module 7 Project Assignment/views

public /Users/praveenkumarsiddela/Downloads/Module 7 Project Assignment/public

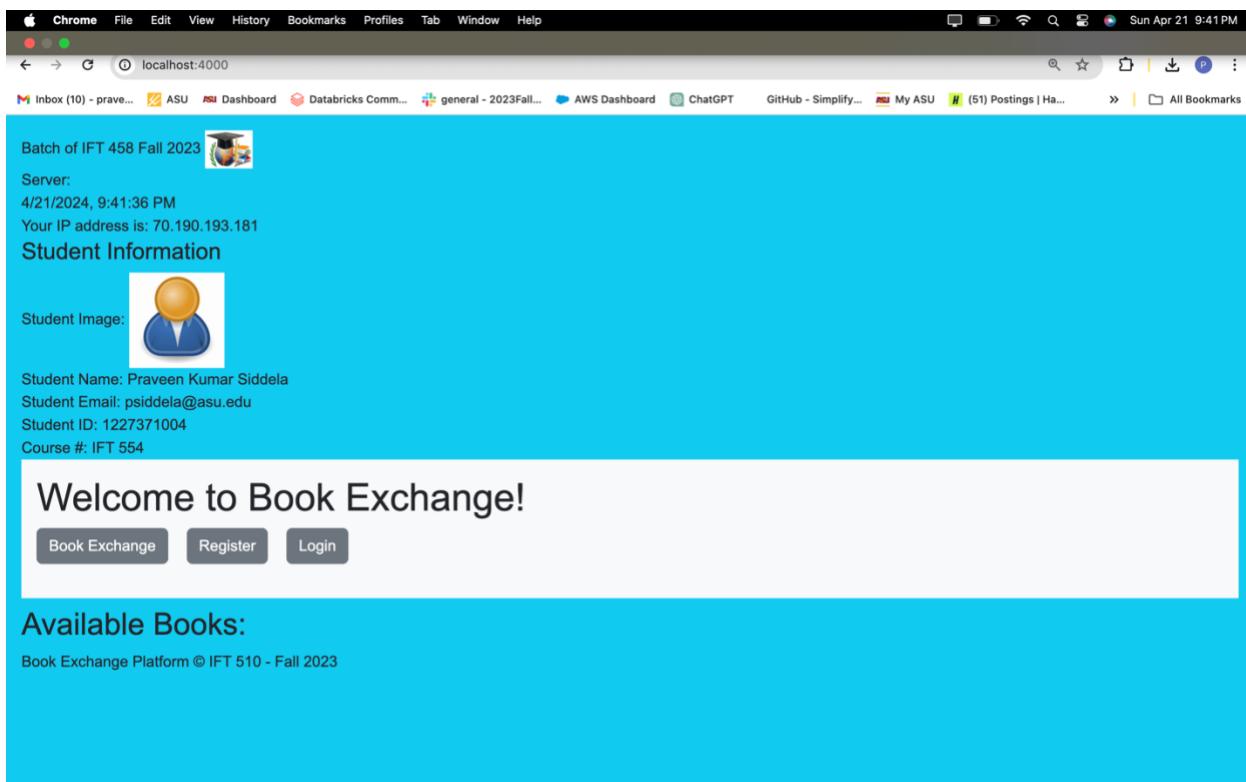
mongodb+srv://praveenkumarsiddela:Praveen5@3a1.u0kjwhr.mongodb.net/?retryWrites=true&w=majority&appName=M3A1

App running on port 4000...

To test the IFT 458 REST App Click Or Type: <http://localhost:4000>...

DB connection successful!

## Home Page:



Batch of IFT 458 Fall 2023 

Server: 4/21/2024, 9:41:36 PM

Your IP address is: 70.190.193.181

**Student Information**

Student Image: 

Student Name: Praveen Kumar Siddela

Student Email: psiddela@asu.edu

Student ID: 1227371004

Course #: IFT 554

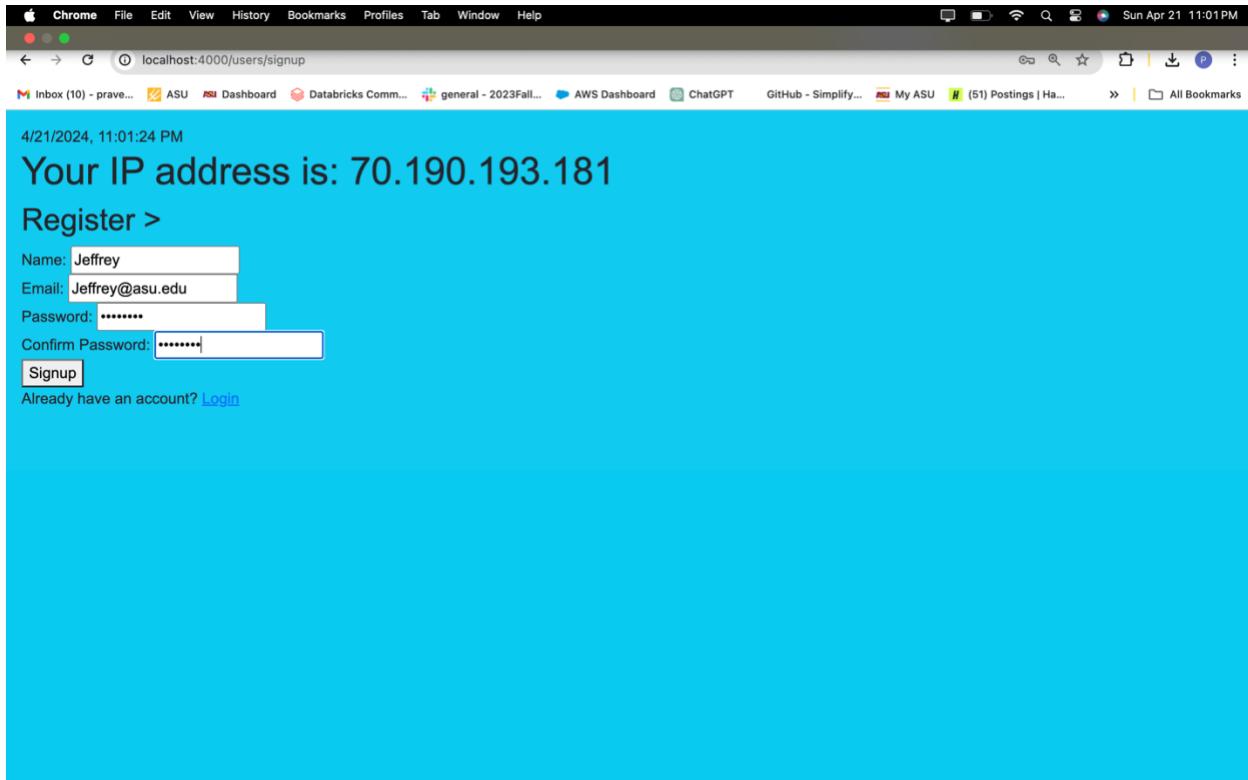
**Welcome to Book Exchange!**

[Book Exchange](#) [Register](#) [Login](#)

**Available Books:**

Book Exchange Platform © IFT 510 - Fall 2023

## Signup:



4/21/2024, 11:01:24 PM

Your IP address is: 70.190.193.181

Register >

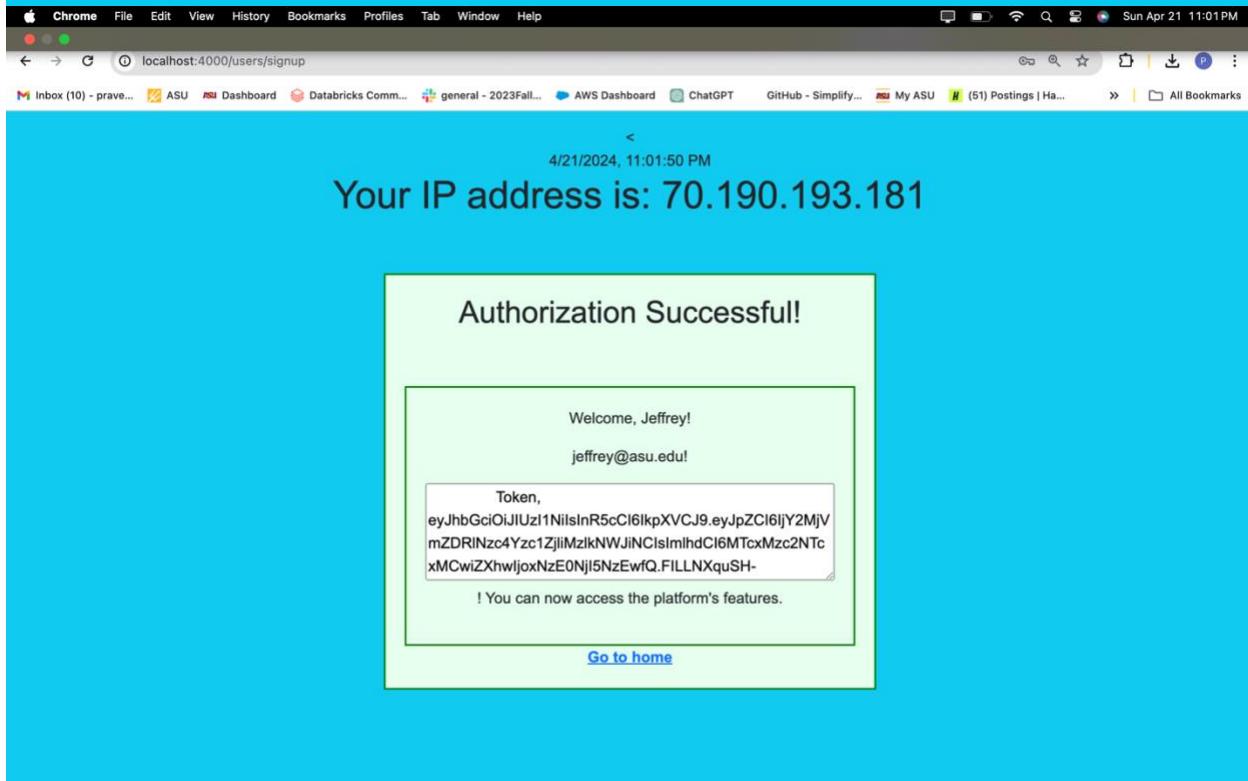
Name:

Email:

Password:

Confirm Password:

Already have an account? [Login](#)

4/21/2024, 11:01:50 PM

Your IP address is: 70.190.193.181

**Authorization Successful!**

Welcome, Jeffrey!

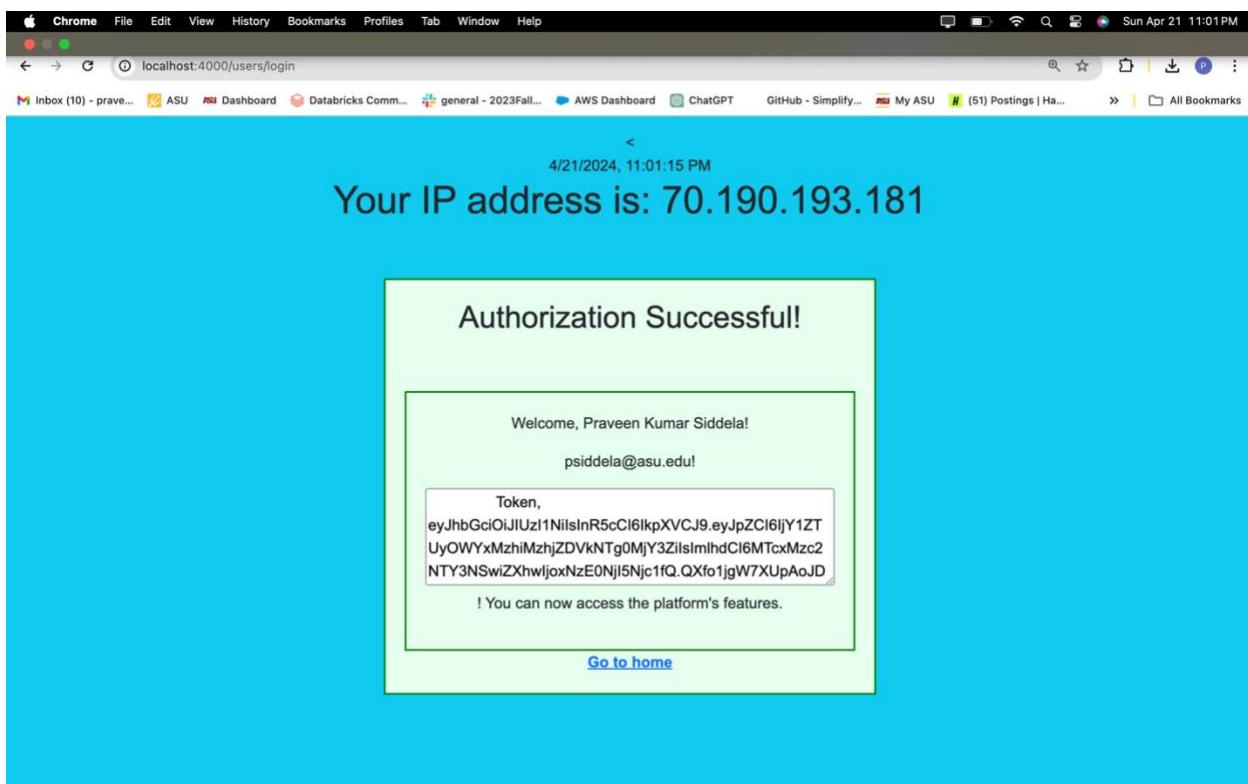
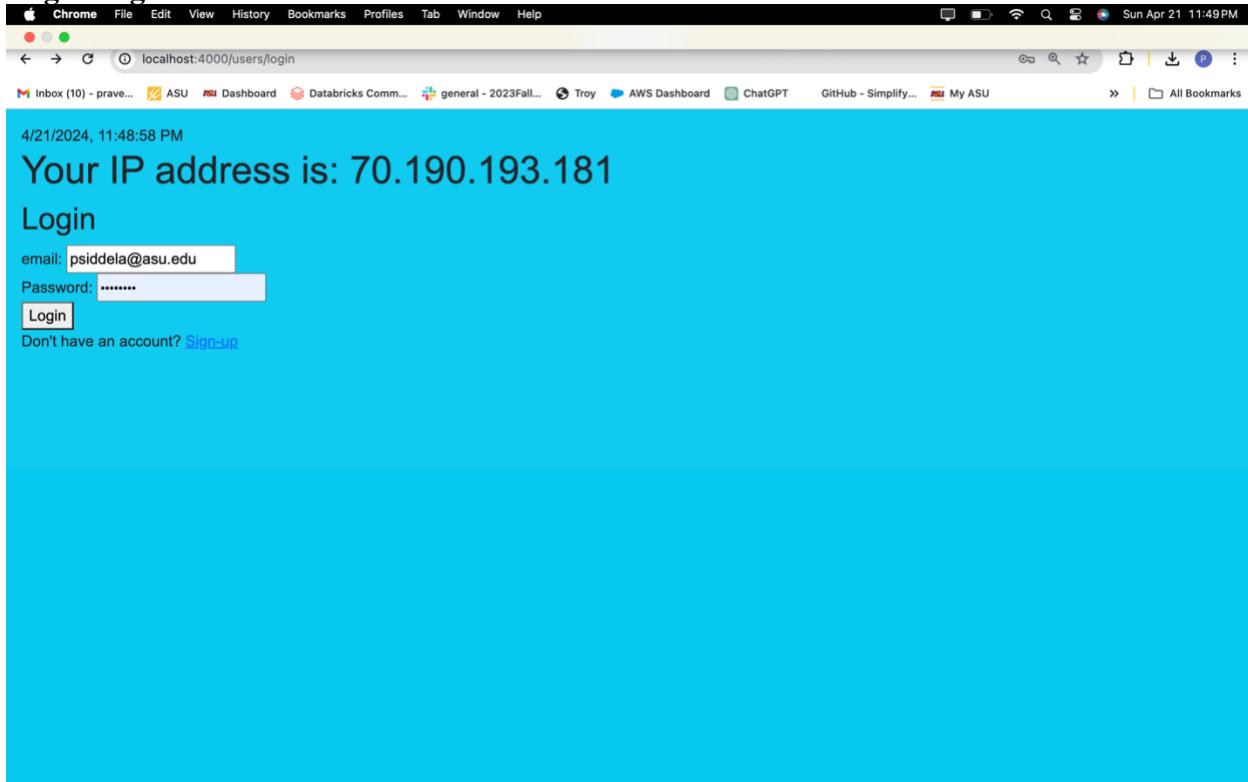
jeffrey@asu.edu!

Token,  
 eyJhbGciOiJIUzI1NilsInR5cCl6IkpXVCJ9.eyJpZCI6IjY2MjVmZDRINzc4Yzc1ZjliMzlkNWJlNCIsImIhdCI6MTcxMzc2NTcxMCwiZXhwIjoxNzE0NjI5NzEwfQ.FILLNXquSH-

! You can now access the platform's features.

[Go to home](#)

## Login Page:



## Retrieve All Books:

4/21/2024, 11:04:12 PM

# Your IP address is: 70.190.193.181

## Book List

Add New Book

| Title                                 | Author       | Description | Exchange Type | Status  | Actions |
|---------------------------------------|--------------|-------------|---------------|---|---------|
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             | available     | <a href="#">Edit</a>   <a href="#">Delete</a> |         |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             | available     | <a href="#">Edit</a>   <a href="#">Delete</a> |         |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             | available     | <a href="#">Edit</a>   <a href="#">Delete</a> |         |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             | available     | <a href="#">Edit</a>   <a href="#">Delete</a> |         |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             | available     | <a href="#">Edit</a>   <a href="#">Delete</a> |         |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             | available     | <a href="#">Edit</a>   <a href="#">Delete</a> |         |

[Go to home](#)

## Add New Book:

4/21/2024, 11:05:51 PM

# Your IP address is: 70.190.193.181

## Add New Book

Title:

Author:

Description:

Exchange Type:

Status:

[Add Book](#)

Chrome File Edit View History Bookmarks Profiles Tab Window Help Sun Apr 21 11:07PM

localhost:4000/books/newBookForm

4/21/2024, 11:05:51 PM

Your IP address is: 70.190.193.181

## Add New Book

Title:

Author:

Description:

Exchange Type:

Status:

[Add Book](#)

Chrome File Edit View History Bookmarks Profiles Tab Window Help Sun Apr 21 11:08PM

localhost:4000/books

4/21/2024, 11:08:14 PM

Your IP address is: 70.190.193.181

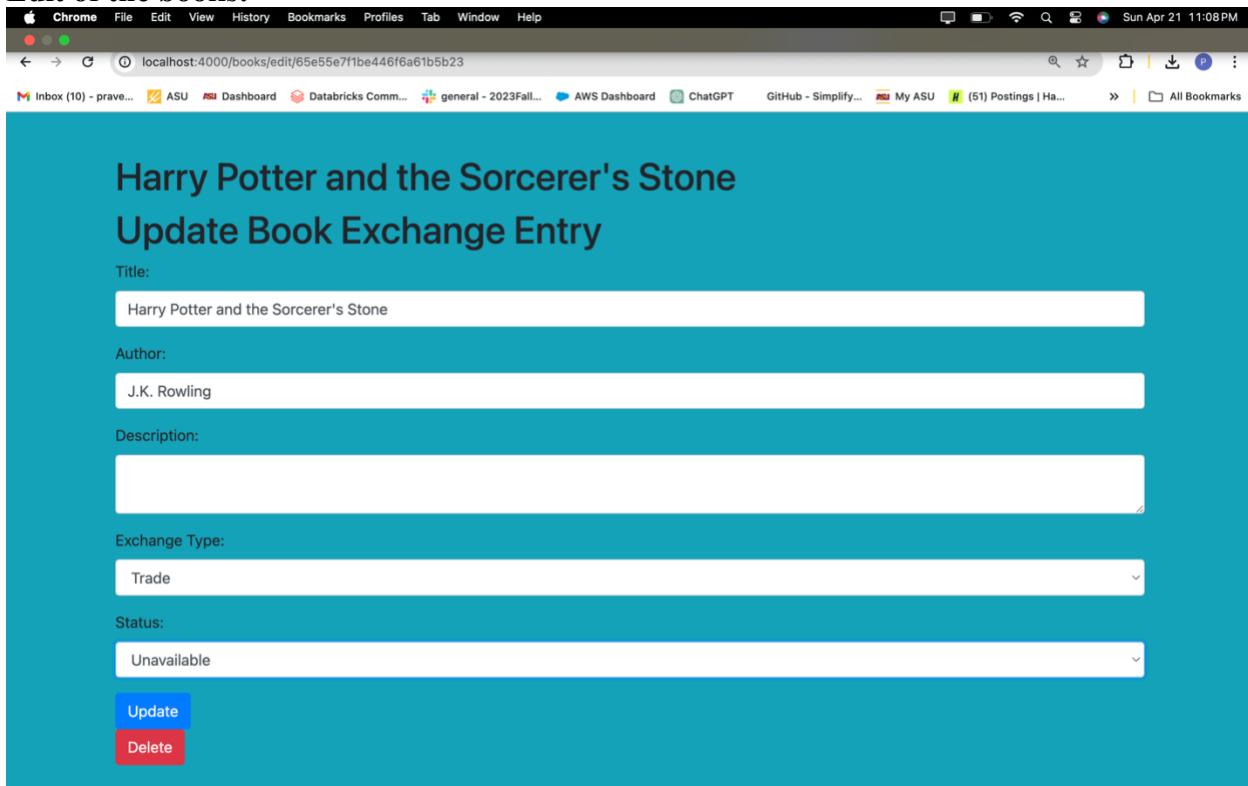
## Book List

Add New Book

| Title                                 | Author       | Description         | Exchange Type | Status    | Actions                                       |
|---------------------------------------|--------------|---------------------|---------------|-----------|---|
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Troy                                  | John Lydgate | Its war book borrow |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |

[Go to home](#)

## Edit of the books:



Harry Potter and the Sorcerer's Stone

Update Book Exchange Entry

Title: Harry Potter and the Sorcerer's Stone

Author: J.K. Rowling

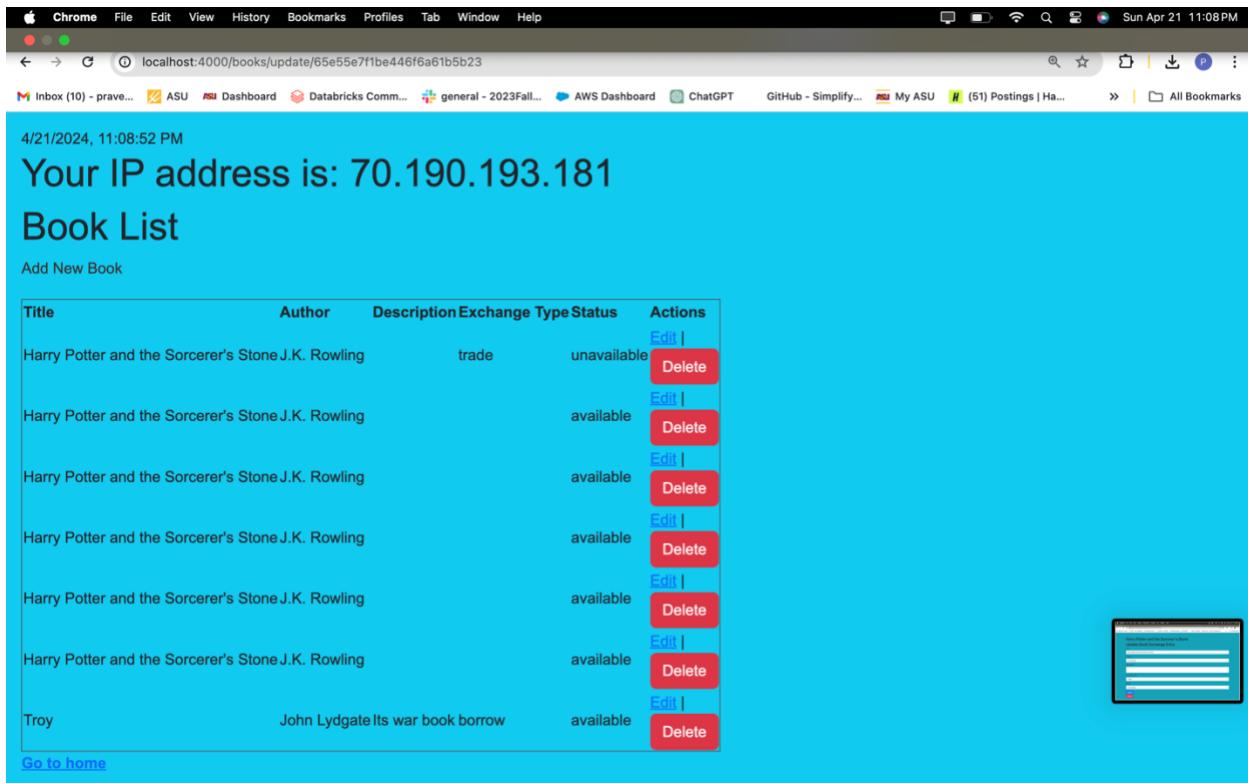
Description:

Exchange Type: Trade

Status: Unavailable

[Update](#) [Delete](#)

## Deletion of Books:



4/21/2024, 11:08:52 PM

Your IP address is: 70.190.193.181

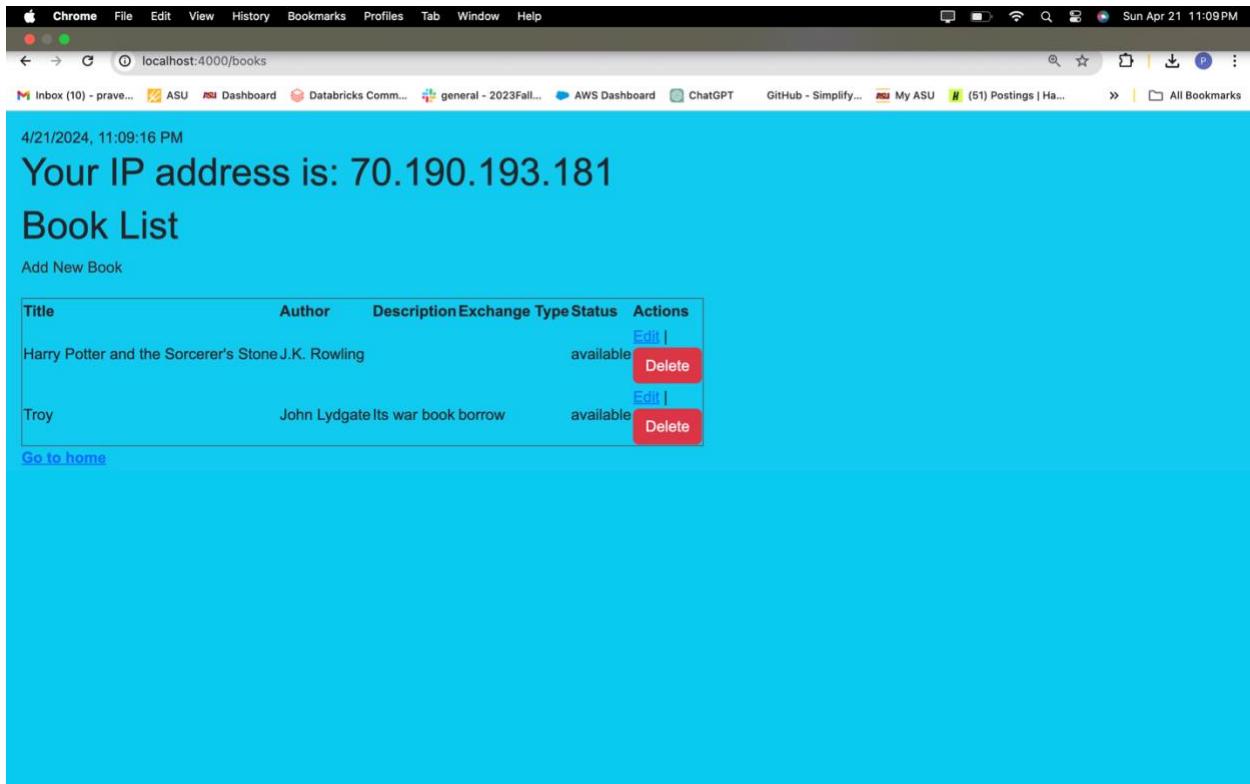
Book List

Add New Book

| Title                                 | Author       | Description         | Exchange Type | Status      | Actions                                       |
|---------------------------------------|--------------|---------------------|---------------|-------------|---|
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     | trade         | unavailable | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available   | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available   | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available   | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available   | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available   | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Troy                                  | John Lydgate | Its war book borrow |               | available   | <a href="#">Edit</a>   <a href="#">Delete</a> |

[Go to home](#)

## After Deletion:



The screenshot shows a Chrome browser window with the URL `localhost:4000/books`. The page displays a "Book List" with two entries:

| Title                                 | Author       | Description         | Exchange Type | Status    | Actions                                       |
|---------------------------------------|--------------|---------------------|---------------|-----------|---|
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |                     |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Troy                                  | John Lydgate | Its war book borrow |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |

Below the table, there is a link [Go to home](#). The browser's address bar shows `localhost:4000/books`. The status bar at the bottom right indicates `Sun Apr 21 11:09:16 PM`.

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

| Feature                | Method | URL End Point (the following are just the sample)<br>You must update the URLs with the correct endpoints)                         |
|------------------------|--------|---|
| Register / Sign up     | POST   | <a href="https://localhost:4000/users/signup">https://localhost:4000/users/signup</a>   |
| Login                  | POST   | <a href="https://localhost:4000/users/login">https://localhost:4000/users/login</a>   |
| Protected Resource     | GET    | <a href="https://localhost:4000/books">https://localhost:4000/books</a>   |
| View All Books         | GET    | <a href="https://localhost:4000/books">https://localhost:4000/books</a>   |
| Add Book Exchange      | POST   | <a href="https://localhost:4000/books/newBookForm">https://localhost:4000/books/newBookForm</a>                                   |
| Update Delete Exchange | POST   | <a href="http://localhost:4000/books/edit/65e565c188ab31ac49627500">http://localhost:4000/books/edit/65e565c188ab31ac49627500</a> |
|                        |        |   |

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

**Post:Adding a New Book**

Code File Edit Selection View Go Run Terminal Window Help

Response(10ms) — Module 7 Project Assignment

```

 1  ###get all blocks
 2  GET http://localhost:4000/books
 3
 4  ##Getting by ID
 5  GET http://localhost:4000/books/65e55e7f1be446f6a61b5b2
 6
 7  ##posting a new block
 8  POST http://localhost:4000/books
 9  Content-Type: application/json
10
11 {
12   "title": "Harry Potter and the Sorcerer's Stone",
13   "author": "J.K. Rowling",
14   "genre": "Fantasy",
15   "year": 2012
16 }

```

HTTP/1.1 200 OK  
X-Powered-By: Express  
Content-Type: text/html; charset=utf-8  
Content-Length: 2298  
ETag: W/"8fa-pE00sYp1fuTDuKx2Ls0d4G3+j40"  
Date: Mon, 22 Apr 2024 06:26:05 GMT  
Connection: close  
<!DOCTYPE html>  
<html>  
<head>  
<title>Login</title>  
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2zHGoQlJyAqSDIyjtNuW8jS1QU4+4oI6Jd48HviuXa4u2F1Q+eXJXZIa0jZ" crossorigin="anonymous">  
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBi4zF7kYXH/4Q+QlW9YHv8+P8t53+qZoZl93CjqtKVEoV5S1f" crossorigin="anonymous"></script>  
<style>  
/\* Add some basic styles for readability \*/  
body {  
font-family: Arial, sans-serif;  
padding: 20px;  
}  
.book {  
border: 1px solid #ccc;  
padding: 10px;  
margin: 10px 0;  
}  
nav {  
margin-bottom: 20px;  
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

mongodb+srv://praveenkumarsiddelaasu:Praveen5@3a1.u0kjwhr.mongodb.net/?retryWrites=true&w=majority&appName=M3A1  
App running on port 4000...  
To test the IFT 458 REST App Click Or Type: http://localhost:4000...  
DB connection successful!

Code File Edit Selection View Go Run Terminal Window Help

Response(4ms) — Module 7 Project Assignment

```

 1  ###get all blocks
 2  GET http://localhost:4000/books
 3
 4  ##Getting by ID
 5  GET http://localhost:4000/books/edit/6625fec778c759b39d5bba
 6
 7  ##posting a new block
 8  POST http://localhost:4000/books
 9  Content-Type: application/json
10
11 {
12   "title": "Harry Potter and the Sorcerer's Stone",
13   "author": "J.K. Rowling",
14   "genre": "Fantasy",
15   "year": 2012
16 }

```

HTTP/1.1 200 OK  
X-Powered-By: Express  
Content-Type: text/html; charset=utf-8  
Content-Length: 2298  
ETag: W/"8fa-pE00sYp1fuTDuKx2Ls0d4G3+j40"  
Date: Mon, 22 Apr 2024 06:27:31 GMT  
Connection: close  
<!DOCTYPE html>  
<html>  
<head>  
<title>Login</title>  
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2zHGoQlJyAqSDIyjtNuW8jS1QU4+4oI6Jd48HviuXa4u2F1Q+eXJXZIa0jZ" crossorigin="anonymous">  
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBi4zF7kYXH/4Q+QlW9YHv8+P8t53+qZoZl93CjqtKVEoV5S1f" crossorigin="anonymous"></script>  
<style>  
/\* Add some basic styles for readability \*/  
body {  
font-family: Arial, sans-serif;  
padding: 20px;  
}  
.book {  
border: 1px solid #ccc;  
padding: 10px;  
margin: 10px 0;  
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

createdAt: 2024-04-22T06:08:13.821Z,  
updatedAt: 2024-04-22T06:08:13.821Z,  
\_\_v: 0

Code File Edit Selection View Go Run Terminal Window Help

Response(4ms) — Module 7 Project Assignment

```

 1  ###get all blocks
 2  GET http://localhost:4000/books
 3
 4  ##Getting by ID
 5  GET http://localhost:4000/books/edit/6625fec778c759b39d5bba
 6
 7  ##posting a new block
 8  POST http://localhost:4000/books
 9  Content-Type: application/json
10
11 {
12   "title": "Harry Potter and the Sorcerer's Stone",
13   "author": "J.K. Rowling",
14   "genre": "Fantasy",
15   "year": 2012
16 }

```

HTTP/1.1 200 OK  
X-Powered-By: Express  
Content-Type: text/html; charset=utf-8  
Content-Length: 2298  
ETag: W/"8fa-pE00sYp1fuTDuKx2Ls0d4G3+j40"  
Date: Mon, 22 Apr 2024 06:27:31 GMT  
Connection: close  
<!DOCTYPE html>  
<html>  
<head>  
<title>Login</title>  
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2zHGoQlJyAqSDIyjtNuW8jS1QU4+4oI6Jd48HviuXa4u2F1Q+eXJXZIa0jZ" crossorigin="anonymous">  
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBi4zF7kYXH/4Q+QlW9YHv8+P8t53+qZoZl93CjqtKVEoV5S1f" crossorigin="anonymous"></script>  
<style>  
/\* Add some basic styles for readability \*/  
body {  
font-family: Arial, sans-serif;  
padding: 20px;  
}  
.book {  
border: 1px solid #ccc;  
padding: 10px;  
margin: 10px 0;  
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

createdAt: 2024-04-22T06:08:13.821Z,  
updatedAt: 2024-04-22T06:08:13.821Z,  
\_\_v: 0

No Environment

4/21/2024, 11:05:51 PM

# Your IP address is: 70.190.193.181

## Add New Book

Title:

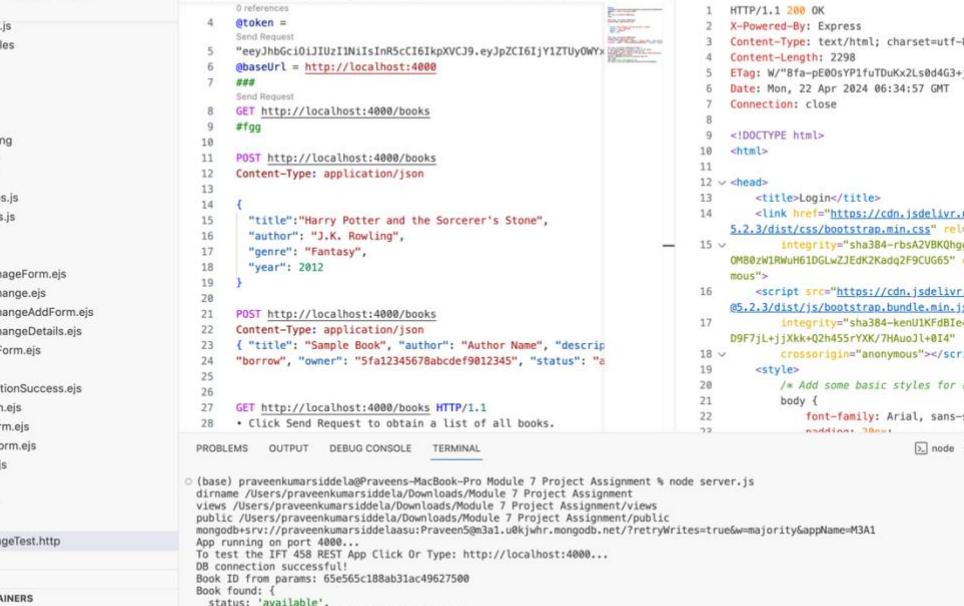
Author:

Description:

Exchange Type:

Status:

## Get Request: Retrieve all Books



The screenshot shows a browser-based code editor with the following interface elements:

- Left Sidebar:** Shows the project structure for "MODULE 7 PROJECT ASSIGNMENT". It includes sections for "models" (userModel.js), "public" (css, img, users, favicon.png, logo.png), "routes" (bookRoutes.js, userRoutes.js), and "views" (books, bookExchangeForm.ejs, bookExchange.ejs, bookExchangeAddForm.ejs, bookExchangeDetails.ejs, bookListForm.ejs), "login" (authorizationSuccess.ejs, loginForm.ejs, logoutForm.ejs), and "register" (registerForm.ejs, appError.ejs, home.ejs, X-UnitTests, app.js, BookExchangeTest.http).
- Central Area:** A code editor window titled "Response(3ms) — Module 7 Project Assignment" showing the "BookExchangeTest.http" file content. The file contains a series of HTTP requests and responses. The responses are as follows:

  - Line 1: HTTP/1.1 200 OK
  - Line 2: X-Powered-By: Express
  - Line 3: Content-Type: text/html; charset=utf-8
  - Line 4: Content-Length: 2298
  - Line 5: ETag: W/"Bfa-#E060Yp1fDuKx2Ls0d4G3+j40"
  - Line 6: Date: Mon, 22 Apr 2024 06:34:57 GMT
  - Line 7: Connection: close
  - Line 8:
  - Line 9: <!DOCTYPE html>
  - Line 10: <html>
  - Line 11:
  - Line 12: <head>
  - Line 13: <title>Login</title>
  - Line 14: <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2ZBKQhggwzxH7pCaAq04GMgnOMBwJlRMWUH61DGLwZJEK2Kadq2F9UG65" crossorigin="anonymous" mouseover>
  - Line 15: <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ken1UKF0B1e4zVFB0sG1M5b4hpcxyD9F7jL+jKKh+Q2h455YXH7HAu0J+814" crossorigin="anonymous">
  - Line 16: <style> /\* Add some basic styles for readability \*/
  - Line 17: body { font-family: Arial, sans-serif; }
  - Line 18:
  - Line 19:
  - Line 20:
  - Line 21:
  - Line 22:
  - Line 23:
  - Line 24:
  - Line 25:
  - Line 26:
  - Line 27: GET http://localhost:4000/books HTTP/1.1
  - Line 28: • Click Send Request to obtain a list of all books.

- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL.
- Terminal:** Shows the command "node BookExchangeTest.http" and its output, which includes the EJS template for the login page and the MongoDB connection details.

4/21/2024, 11:04:12 PM

Your IP address is: 70.190.193.181

## Book List

Add New Book

| Title                                 | Author       | Description | Exchange Type | Status    | Actions                                       |
|---------------------------------------|--------------|-------------|---------------|-----------|---|
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Harry Potter and the Sorcerer's Stone | J.K. Rowling |             |               | available | <a href="#">Edit</a>   <a href="#">Delete</a> |

[Go to home](#)

### GET Request: Fetch Specific Book by ID:

Code File Edit Selection View Go Run Terminal Window Help

Response(5ms) — Module 7 Project Assignment

```

 5 "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpczI6IjY1ZTUY0NyA
 6 <baseURL = http://localhost:4000
 7 /**
 8   Send Request
 9   #f99
10
11 POST http://localhost:4000/books
12 Content-Type: application/json
13
14 {
15   "title": "Harry Potter and the Sorcerer's Stone",
16   "author": "J.K. Rowling",
17   "genre": "Fantasy",
18   "year": 2012
19 }
20
21 POST http://localhost:4000/books
22 Content-Type: application/json
23 { "title": "Sample Book", "author": "Author Name", "descrip
24 "borrow", "owner": "5fa12345678abcdef9012345", "status": "a
25
26
27 GET http://localhost:4000/books HTTP/1.1
28 Click Send Request to obtain a list of all books.
29 5. GET Request: Fetch Specific Book by ID:
30 Requirements: Acquire 3 books by ID and submit before-and-a

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

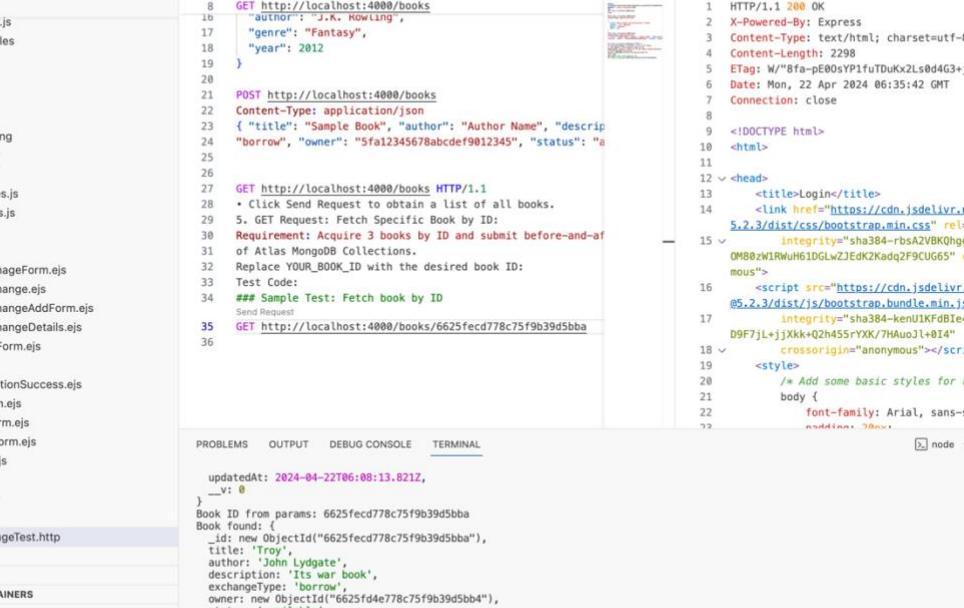
```

(base) praveenkumarreddela@Praveen's-MacBook-Pro:~/Module 7 Project Assignment% node server.js
dls@Praveen:~/Module 7 Project Assignment$ node server.js
views /Users/praveenkumarreddela/Downloads/Module 7 Project Assignment/views
public /Users/praveenkumarreddela/Downloads/Module 7 Project Assignment/public
mongodb://praveenkumarreddela:Praveen5@3b1.u0kjwhr.mongodb.net/?retryWrites=true&w=majority&appName=M3A1
App running on port 4000...
To test the IFT 458 REST App Click Or Type: http://localhost:4000...
DB connection successful!
Book found from params: 65e565c188ab31ac49627580
Book found:
  status: 'available',
  _id: new ObjectId("65e565c188ab31ac49627580"),
  title: "Harry Potter and the Sorcerer's Stone",
  author: 'J.K. Rowling',
  genre: 'Fantasy',
  year: 2012,
  createdAt: 2024-03-04T06:10:09.395Z,

```

No Environment

## PUT Request: Modify a Specific Book by ID:



The screenshot shows a Mac OS X desktop with a browser window in the foreground displaying a Node.js project structure. The browser title is "Response(4ms) — Module 7 Project Assignment". The project structure on the left includes "MODULE 7 PROJECT ASSIGNMENT", "node\_modules", "public", "img", "users", "routes", and "views" with subfolders like "books" and "exchange". A file "app.js" is also listed. The "BOOKEXCHANGETEST.HTTP" file is selected, showing code for a BookExchangeTest. The code includes GET and POST requests to "localhost:4000/books" and a sample test for fetching a book by ID. The browser's right sidebar shows the response content, which includes the book details and a rendered HTML page for a login form. The bottom of the browser window shows tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with the TERMINAL tab active. The terminal output shows the book details and a rendered HTML page for a login form.

```
Code File Edit Selection View Go Run Terminal Window Help
Response(4ms) — Module 7 Project Assignment
BOOKEXCHANGETEST.HTTP
8  GET http://localhost:4000/books
1b   "author": "J.K. ROWLING",
17   "genre": "Fantasy",
18   "year": 2012
19 }
20
21 POST http://localhost:4000/books
22 Content-Type: application/json
23 { "title": "Sample Book", "author": "Author Name", "descrip
24 "borrow", "owner": "5fa12345678abcdef9012345", "status": "a
25
26
27 GET http://localhost:4000/books HTTP/1.1
28 * Click Send Request to obtain a list of all books.
29 5. GET Request: Fetch Specific Book by ID:
30 Requirement: Acquire 3 books by ID and submit before-and-aft
31 of Atlas MongoDB Collections.
32 Replace YOUR_BOOK_ID with the desired book ID:
33 Test Code:
34 ### Sample Test: Fetch book by ID
35 GET http://localhost:4000/books/6625fecd778c75f9b39d5bba
36

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
updatedAt: 2024-04-22T06:08:13.821Z,
} _v: 0
Book ID from params: 6625fecd778c75f9b39d5bba
Book found: {
  _id: new ObjectId("6625fecd778c75f9b39d5bba"),
  title: 'Troy',
  author: 'John Lydgate',
  description: 'its own book',
  exchangeType: 'borrow',
  owner: new ObjectId("6625fd4e778c75f9b39d5bba"),
  status: 'available',
  createdAt: 2024-04-22T06:08:13.821Z,
  updatedAt: 2024-04-22T06:08:13.821Z,
} _v: 0
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 2298
5 ETag: W/"8fa-E005yP1fTuDkx2Ls0dG3+j40"
6 Date: Mon, 22 Apr 2024 06:35:42 GMT
7 Connection: close
8
9 <!DOCTYPE html>
10 <html>
11 <head>
12   <title>Login</title>
13   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2ZHBQJbzJEDKXadqZFCU6G5" crossorigin="anonymous">
14   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KfdB1e4zVFB0s0G1M5b4hcpxyD9F7jL+jXkk+Q2h455YXK/7HauJL+014" crossorigin="anonymous"></script>
15   <style>
16     /* Add some basic styles for readability */
17     body {
18       font-family: Arial, sans-serif;
19       padding: 20px;
20     }
21   </style>
22 
```

## Learning Outcomes of Module 4 to Module 7

### Technical Learning Outcomes:

#### Module 4

In web applications, authentication and authorization are pivotal for ensuring secure user interactions. Authentication verifies users' identities using credentials like usernames and passwords, while authorization determines the actions users are allowed to take post-authentication. Session management is equally critical, as it maintains stateful information about user interactions, encompassing the creation, storage, and secure disposal of session data. Token-based authentication is a prevalent method, issuing tokens to authenticated users for subsequent authorization requests, with JSON Web Tokens (JWT) commonly employed due to their secure and efficient data transmission.

Testing RESTful APIs is imperative to guaranteeing their reliability and functionality. Integrated Development Environment (IDE) tools like Visual Studio Code offer REST client plugins, enabling direct API testing with simple syntax for interacting with various endpoints using standard HTTP methods. Additionally, debugging Node.js applications is vital for swiftly identifying and rectifying errors. IDEs provide debugging tools allowing developers to set breakpoints, pause code execution, and inspect variables and states, streamlining the debugging process for efficient troubleshooting. Protecting application resources involves implementing robust access controls and token expiry mechanisms, alongside session persistence techniques like cookies or local storage, ensuring secure user interactions and seamless experiences.

## Module 5

Installing Git and Visual Studio Code is the first step in establishing a robust development environment, enabling version control and efficient project management. Cloning repositories allows developers to create local copies of remote codebases, fostering local development and facilitating collaboration with team members. Making changes to code, whether it involves editing existing files, adding new features, or removing unnecessary code, is essential for project progression and improvement. Staging and committing changes with descriptive messages ensures an organized version history, aiding in tracking project evolution and facilitating collaboration among team members.

Pushing changes to the remote repository enables synchronization between local and remote codebases, ensuring seamless collaboration and sharing of progress across the team. Branching out from the main codebase allows developers to isolate work on specific features or fixes, promoting development flexibility and minimizing conflicts. Middleware implementation in Node.js intercepts and processes incoming requests, facilitating tasks such as logging and authentication, thereby enhancing the functionality and security of web applications. Finally, opening pull requests initiates discussions, facilitates code review, and integrates proposed changes into the main repository, promoting collaboration and ensuring code quality and consistency across the project.

## Module 6

Chained middleware enables the sequential execution of multiple middleware functions, allowing for modular and organized request processing in Node.js applications. Custom middleware development is crucial for addressing specific tasks such as logging and authentication, tailoring middleware functions to meet the application's unique requirements. Middleware error handling

plays a vital role in enhancing application reliability by implementing middleware to catch errors and send appropriate responses, ensuring smooth user experiences and preventing potential system failures. Verifying Docker, Node.js, and Visual Studio Code installations is essential for setting up the development environment, ensuring compatibility and readiness for containerization and application development processes. Writing a Docker file and building Docker images within Visual Studio Code streamlines the process of Docker image creation, facilitating efficient deployment and management of containerized applications. Running Docker containers and configuring settings allows developers to manage containerization effectively, ensuring optimal performance and resource utilization while accessing the deployed Node.js app.

Handling dynamic parameters in routes through dynamic routing mechanisms enhances the flexibility and scalability of web applications, accommodating varying user inputs and requirements. Middleware for request validation is crucial for ensuring data integrity and security by validating URL parameters and responding to errors appropriately, safeguarding against malicious inputs and potential vulnerabilities. Implementing complex middleware chains tailored to route-specific tasks allows for intricate request processing and handling in Node.js applications, enabling robust and efficient application behavior. Lastly, error handling middleware plays a crucial role in managing various error types, ensuring graceful degradation and consistent performance even in the face of unexpected issues, thereby enhancing the reliability and resilience of the application.

## **Module 7**

To ensure secure communication in a Windows environment, OpenSSL installation is essential, providing a toolkit for SSL/TLS protocols. Generating a self-signed SSL certificate and private key using OpenSSL enables secure connections for applications. Configuring an HTTPS server in

a Node.js application with the generated SSL certificate and private key ensures encrypted communication, enhancing data security. Implementing HTTP status codes like 200, 201, 400, 404, and 500 allows for precise server responses, improving user experience and troubleshooting. Developing custom error handling middleware, including 404 (Not Found) and general server errors, ensures robust error management and graceful degradation. Using try/catch blocks with Promises facilitates graceful error handling during asynchronous operations, enhancing application reliability.

For a secure middleware and backend database setup in a Book Exchange Application, organizing the project directory structure according to the Model-View-Controller (MVC) pattern ensures a clear separation of concerns. Reviewing the book model schema and relationships for storing book information in the backend database provides insights into data organization. Implementing logic for CRUD operations related to books in the backend through the book controller enhances data management and manipulation capabilities. Defining routes for handling HTTP requests related to book resources establishes endpoints for CRUD operations, facilitating communication between the frontend and backend. Customizing view templates (\*.ejs files) in the view folder personalizes the user interface and enables interaction with backend APIs. Configuring database connections, starting the server, and testing API functionality using tools like Postman or Thunder Client ensures proper application execution and functionality verification.

## References

<https://medium.com/@dushyanthak/best-practices-for-writing-custom-middlewares-in-asp-.net-core-97b58c50cf9c>

<https://stackoverflow.com/questions/62398/what-are-the-best-practices-for-the-middleware-api>

<https://www.omnitas.com/how-to-choose-the-right-middleware-for-enterprise-integration/>

## Appendices

### 1. Sample Code Snippets:

- Views (bookExchange.ejs)
- Server (server.ejs)
- Controller(booksController.js)

### 2. Screenshots:

- Updated view files (\*.ejs) and changes made
- BookModel.js file and explanation
- Home page displayed in the browser
- Personalized user interface pages
- Testing process and responses for different API endpoints captured in Postman
- Project directory structure showcasing MVC organization
- Architecture diagram showcasing the components and their interactions
- Flowchart illustrating the flow of the Book Exchange Application
- Terminal showing successful installation of dependencies
- bookController.js with implemented logic
- bookRoutes.js file with defined routes

### 3. Configuration Files:

Configuration file (config.env) containing database connections and JWT parameters.  
Server.js script setting up an HTTPS server in Node.js.

### 4. API Specification:

- API documentation detailing the endpoints, request methods, and expected data formats.

- Table presenting the Book Exchange API Specifications, containing endpoint specifics and descriptions gathered during testing.