

## **Book Exchange Website**

Sushmita Prafull Halasawade

IFT 544: Middleware Prog & Database Sec

Dinesh Sthapit

1 Dec 2023

## Table of Contents

<b>Abstract:</b> .....	<b>3</b>
<b>Introduction:</b> .....	<b>3</b>
Overview of the project: .....	3
Problem Statement: .....	4
Scope and Limitations: .....	4
Technology Stack:.....	4
<b>Project Design:</b> .....	<b>5</b>
Architecture and Design:.....	5
Security Flow: .....	6
Entity Relationship Diagram (ERD):.....	8
MVC Architecture:.....	10
API specification: .....	14
<b>Screenshots:</b> .....	<b>29</b>
<b>Implementation:</b> .....	<b>44</b>
Challenges Faced During Implementation: .....	44
Workarounds and Solutions Applied:.....	45
Key Functionalities and Features: .....	45
<b>Learning Outcomes:</b> .....	<b>46</b>
<b>Constructive Feedback:</b> .....	<b>47</b>
<b>References:</b> .....	<b>49</b>

## Abstract:

The "Book Exchange" website is a user-friendly online platform designed for book enthusiasts to exchange books with each other. Developed using Node.js, Express, and MongoDB, this website offers a seamless user experience with a clean and responsive interface. Its primary functionalities include user authentication, book management, and handling exchange requests. Users can easily sign up, log in, list books for exchange, browse books offered by others, and initiate exchanges. The project architecture is robust, featuring well-structured modules for user and book management. Each user's profile contains essential information such as name, email, and encrypted passwords, while the book profiles include details like title, author, and exchange status. Security is a key aspect, with JWTs implemented for secure authentication and bcrypt.js for password hashing. Middleware functions further enhance security by protecting routes that require authentication. The Book Exchange website is a secure, easy-to-use platform that helps people share books and enjoy reading together online.

## Introduction:

### Overview of the project:

The "Book Exchange" website is a new online platform that helps readers exchange books. It's like a digital marketplace where people can list, browse, and exchange books. The goal of this project is to encourage reading and sharing of books by creating a network where book lovers can find and share books without having to buy them.

#### Problem Statement:

The main problem that the "Book Exchange" website solves is that there is no good, easy-to-use platform for book lovers to exchange books. Many readers have books they don't need any more and want to find new books to read. But the options for doing this are often limited, inconvenient, or expensive. This project aims to fix these problems by providing a central, easy-to-use platform specifically for exchanging books.

#### Scope and Limitations:

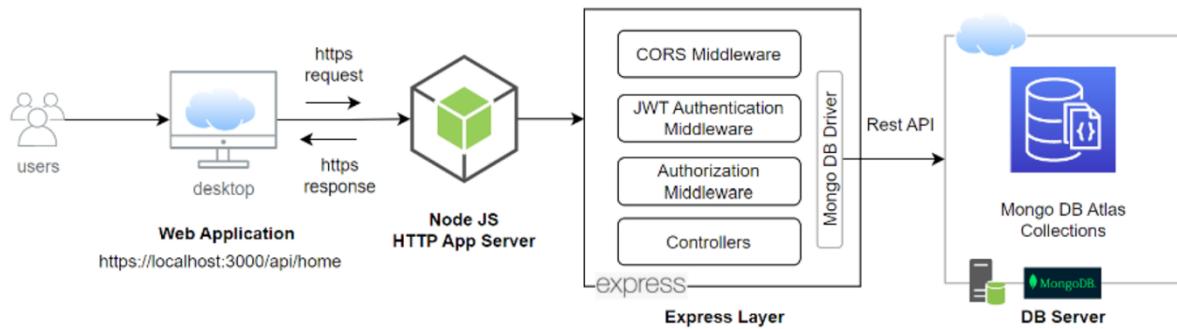
The scope of the project is to develop a full-fledged web application with features for user registration, book listing, and managing exchange requests. However, there are some limitations. The platform currently only works online, so you need internet access to use it. Additionally, the project does not handle the logistics or physical exchange of books, which is left to the users to coordinate. It relies on user honesty and accountability for the condition and delivery of books

#### Technology Stack:

This app uses Node.js, Express, and MongoDB technologies. Node.js provides a scalable environment for handling many connections at the same time, which is important for a community-driven platform. Express, a flexible Node.js web application framework, makes it easy to develop robust web features. MongoDB, a NoSQL database, is flexible and good at handling large amounts of unstructured data, making it suitable for the dynamic and growing collection of user and book information. This technology stack ensures a reliable, scalable, and user-friendly application, which aligns perfectly with the project's needs.

## Project Design:

### Architecture and Design:



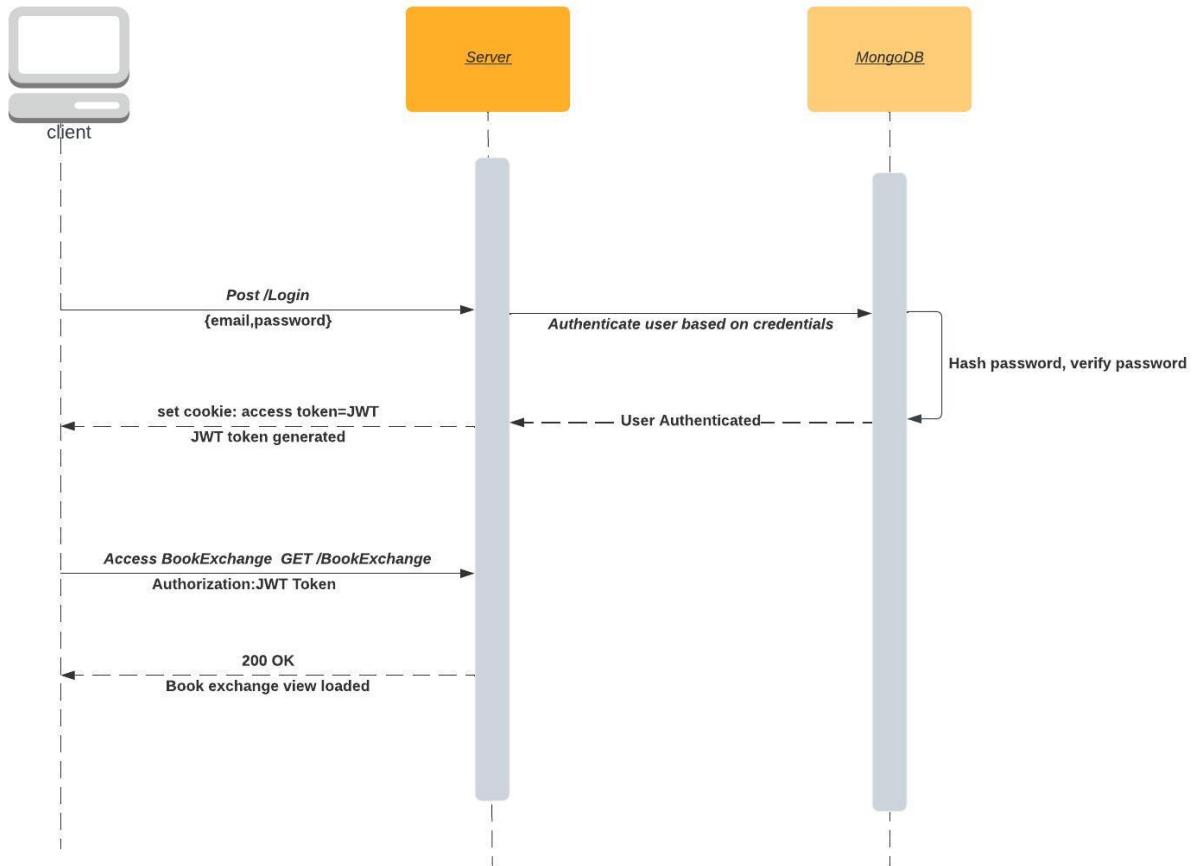
- **User Interaction:** User interacts with the web application through a browser or client-side application.
- **HTTP Request Processing:** The user's HTTP request is received by the Node.js HTTP application server.
- **Express Layer:** The request is then routed to the Express layer, which acts as a middleware framework.
- **Within Express:** CORS middleware handles Cross-Origin Resource Sharing to manage access control; JWT authentication verifies and authenticates user identity based on JSON Web Tokens; Authorization logic determines access rights before proceeding; Controller functions handle the business logic and orchestrate data processing.
- **Database Interaction:** RESTful API requests, formed by the Express layer, are directed to the database server; MongoDB driver facilitates communication between the Node.js application and the database.

application and the MongoDB database. Database operations are performed based on the received RESTful API requests.

- **Data Processing:** The database server processes the received requests, executing operations such as CRUD (Create, Read, Update, Delete) in the MongoDB database.
- **HTTP Response Handling:** After processing, the response is sent back through the same flow. The response travels from the database server to the Node.js HTTP application server.

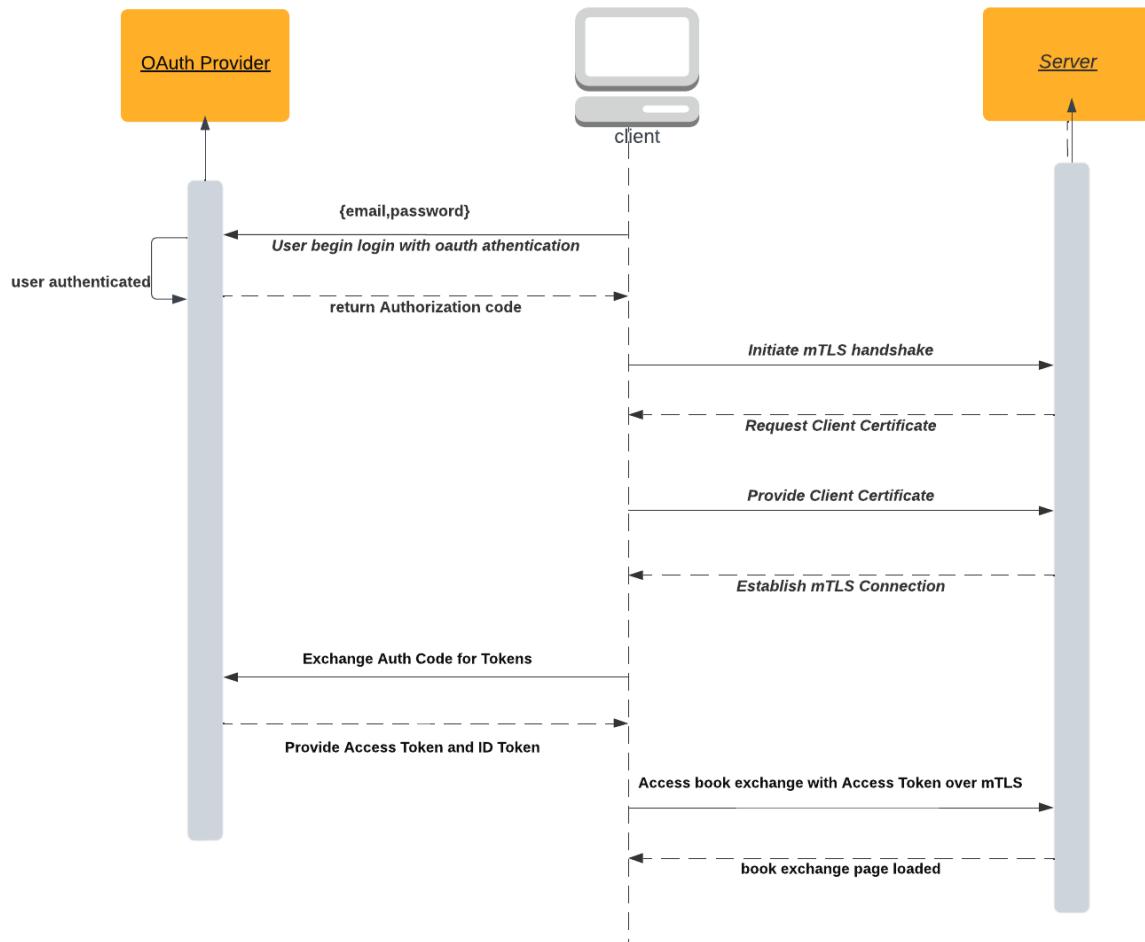
Security Flow:

#### Authentication and authorization with JWT (JSON Web Tokens Introduction, n.d.):



- User logs in with email and password; JWT token is generated and set as a cookie.
- Authenticated user accesses the book exchange page displaying available book details.
- JWT token is sent with each request, allowing the server to authenticate and authorize the user.

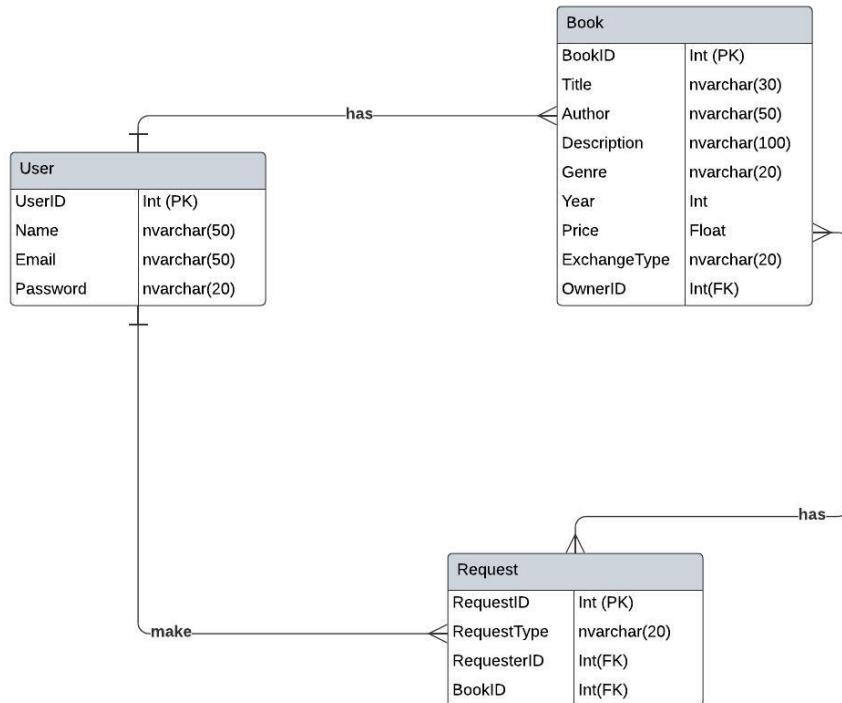
### **Authentication and Authorization process after oauth (oAuth, 2020)and mtls(Porter, 2021):**



- User selects "Login via OAuth Provider" on the client application.
- OAuth server presents authentication interface to the user.

- User enters credentials and authenticates with the OAuth provider.
- OAuth server redirects back to the client with an authorization code.
- Client exchanges authorization code for an access token and an ID token.
- OAuth server sends the tokens back to the client.
- Client initiates a TLS handshake with the server.
- Server requests the client's certificate.
- Client presents its certificate.
- Server authenticates the client's certificate and establishes an mTLS connection.
- Client sends a request to the resource server with the access token over the mTLS connection.
- Resource server validates the token and serves the request.

Entity Relationship Diagram (ERD):



**User Entity:** Represents the registered users of the application. Each user can own multiple books.

Attributes: UserID (Primary Key), Name, Email, Password

Relationships: One-to-Many with Book Entity (One user can have multiple books)

**Book Entity:** Represents the books available for exchange. Each book is associated with a user (owner). It also has a Many-to-Many relationship with the Request Entity, indicating that a book can have multiple requests.

Attributes: BookID (Primary Key), Title ,Author ,Description ,Genre ,Year ,Price, ExchangeType (Borrow/Trade) , OwnerID (Foreign Key referencing UserID from User Entity)

Relationships: Many-to-One with User Entity (Many books can belong to one user), Many-to-Many with Request Entity (Many books can have many requests)

**Request Entity:** Represents the requests made by users for borrowing or trading books. It includes information about the type of request, the requester, and the book being requested.

Attributes: ,RequestID (Primary Key) ,RequestType (Borrow/Trade) ,RequesterID (Foreign Key referencing UserID from User Entity) ,BookID (Foreign Key referencing BookID from Book Entity)

Relationships: Many-to-One with User Entity (Many requests can be made by one user), Many-to-Many with Book Entity (Many requests can be made for one book)

## MVC Architecture:

- **Model:** The models define the data structure. In this project, there are two primary models: User and BookExchange. The User model includes fields like name, email, password, while the BookExchange model has fields for title, author, description, exchange type, owner, and status. These models are defined using Mongoose (Mongoose, 2011), a MongoDB object modeling tool, allowing for easy interaction with the database.

```
bookModel.js
models > bookModel.js > bookExchangeSchema > status
  const mongoose = require('mongoose');

  const bookExchangeSchema = new mongoose.Schema({
    title: {
      type: String,
      required: true
    },
    author: {
      type: String,
      required: true
    },
    exchangeType: {
      type: String,
      enum: ['borrow', 'trade'],
      required: true
    },
    owner: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'User',
      required: true
    },
    status: {
      type: String,
      enum: ['available', 'unavailable'],
      default: 'available'
    },
    timestamps: true
  });

  const BookExchange = mongoose.model('BookExchange', bookExchangeSchema);

  module.exports = BookExchange;
```

```
userModel.js
models > userModel.js > userSchema > name > required
  const mongoose = require('mongoose');

  const userSchema = new mongoose.Schema({
    name: {
      type: String,
      required: [true, 'Please tell us your name!']
    },
    email: {
      type: String,
      required: [true, 'Please provide your email'],
      unique: true,
      lowercase: true
    },
    photo: String,
    role: {
      type: String,
      enum: ['client', 'staff', 'student', 'admin'],
      default: 'client'
    },
    password: {
      type: String,
      required: [true, 'Please provide a password'],
      minlength: 8,
      select: false
    },
    // validate password
    passwordConfirmation: {
      type: String,
      required: [true, 'Please confirm your password'],
      validate: {
        validator: function(pwd) {
          return pwd === this.password;
        },
        message: "the password confirmation did not match"
      }
    },
    passwordChangedAt: Date,
    passwordResetExpires: Date,
    active: {
      type: Boolean,
      default: true,
      select: false
    },
    roles: {
      type: String,
      default: 'client' // authorization
    }
  });

  module.exports = mongoose.model('User', userSchema);
```

- **View:** Views are used to render the user interface. This project uses .ejs (Ejs, 2018) templates for views, which allow embedding JavaScript into HTML templates. These views are rendered based on the data received from the controllers and are displayed to the user. For instance, there are views for listing books, adding new books, and user authentication processes.

```

views > login > loginForm.ejs > html > head > style > .book

<!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-rbsA2VbkOHggwzxi7pPCaQ046Mgn0M802wIRWuh6IDGLwZEdK2Kadq2F9CUG65" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-kenU1KfDbie4zVF0sGJM5b4ncpxyD9F7jl+JXkk+Q2h455rYXk/HAuJl+0I4">
      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;
      }

      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>
    <div>
      <label>Server: </label>
      <div id="currentDateTime"></div>
      <div>Your IP address is: <span id="ip-address"></span></div>
    </div>
    <h4>Student Information</h4>
  </body>

```

Ln 19, Col 36 Spaces: 4 UTF-8 CRLF ⓘ HTML ⌂ Prettier

- **Controller:** Controllers handle the business logic. This codebase includes controllers like authController for handling user authentication and booksController for managing book-related operations. For example, booksController contains functions for adding, creating, updating, and deleting books, as well as fetching book details.

```

JS authController.js ×
controllers > JS authController.js > Signup > Signup
1 const crypto = require('crypto');
2 const User = require('../models/userModel');
3 const jwt = require('jsonwebtoken');
4
5 // since the values are same as the properties names
6 // this style is called object destructuring and it is preferred
7 exports.signup = async (req, res, next) => {
8   const { name, email, password, passwordConfirmation } = req.body;
9   try {
10     const newUser = await User.create({
11       name,
12       email,
13       password,
14       passwordConfirmation
15     });
16     // and then send the token
17     createSendToken(newUser, 201, res); // this is called Authorization
18   } catch (error) {
19     res.status(400).render('../login/registerForm', {
20       errorCode: 400,
21       errorMessage: error.message
22     });
23   }
24 };
25
26 const createSendToken = (user, statusCode, res) => {
27   const token = signToken(user._id);
28   const cookieOptions = {
29     expires: new Date(
30       Date.now() + process.env.JWT_COOKIE_EXPIRES_IN * 24 * 60 * 60 * 1000
31     ),
32     httpOnly: true
33   };
34   if (process.env.NODE_ENV === 'production') cookieOptions.secure = true;
35
36   res.cookie('jwt', token, cookieOptions);
37
38   // Remove password from output
39   user.password = undefined; // very important
40   res.render('../login/authorizationSuccess', { user, token });
41 };
42 // sign token
43 const signToken = id => {
44   // document reference https://www.npmjs.com/package/jsonwebtoken
45   try {
46     const jwtToken = jwt.sign({ id }, process.env.JWT_SECRET, {

```

```

JS booksController.js ×
controllers > JS booksController.js ...
1 const Book = require('../models/bookModel');
2
3 // Render a new book from send the HTML form to the client (browser) so that t
exports.addBook = async (req, res) => {
4
5   try {
6     res.render('../books/bookExchangeAddForm');
7   } catch (err) {
8     res.status(400).render('appError', { title: 'Add new Book', user: req.user });
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 authentication
21       status: req.body.status // Or default to 'available' as per your s
22     });
23     const newBook = await book.save();
24     res.redirect('/books'); // assuming '/books' is where you list all boo
25   } catch (err) {
26     res.status(400).render('appError', { title: 'Create Book', user: req.user });
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 });
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) {
45       return res.status(404).render('error', { message: 'Book not fo
46   }

```

- Routes:** The website's routing is handled using Express, a web application framework for Node.js. Routes are defined to respond to various HTTP requests (GET, POST, PUT, DELETE) corresponding to different functionalities. For instance: Routes related to books (bookRoute) include paths for creating a new book (/newBookForm), adding a book (/add), and getting book details (/:id). User-related routes (userRoute) handle user authentication, including signup (/signup) and login (/login).

```

JS bookRoutes.js ×
routes > JS 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
6 // Authenticate middleware
7 const authenticate = authenticateAuthorize.authenticate;
8
9 // Authorize middleware
10 const authorize = authenticateAuthorize.authorize;
11
12 // Render a new book Form and send the form to the client (browser) so that the user can
13 router.get('/newBookForm', authenticate, authorize, booksController.addBook);
14
15 // CREATE is new book via a POST
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.post('/', authenticate, authorize, booksController.createBook);
19
20 // READ is done via GET
21 // note that the authenticate and authorize middleware are passed as arguments
22 // the node frameworks allows you to pass as many middleware as you want
23 router.get('/', authenticate, authorize, booksController.getBooks);
24
25 // READ: GET a specific book by ID
26 // note that the authenticate and authorize middleware are passed as arguments
27 // the node frameworks allows you to pass as many middleware as you want
28 router.get('/:id', authenticate, authorize, booksController.getBookById);
29
30 // UPDATE is done via PUT (update) an existing book by ID
31 // Update book Form
32 router.get('/edit/:id', authenticate, authorize, booksController.updateBookForm);
33 // UPDATE i update
34 // note that the authenticate and authorize middleware are passed as arguments
35 // the node frameworks allows you to pass as many middleware as you want
36 // The action attribute in your <form> tag specifies the URL to which the form data should
37 // In your case, the form action is set to / books / update /%<=book._id%>.
38 // This will dynamically insert the _id of the book you are updating,
39 // effectively sending a POST request to a URL like / books / update / 12345 (where 12345
40 // post('/update/:id', authenticate, authorize, booksController.updateBookById);
41
42 // DELETE is done via DELETE Command a book by ID
43 // note that the authenticate and authorize middleware are passed as arguments
44 // the node frameworks allows you to pass as many middleware as you want
45 // The action attribute in your <form> tag specifies the URL to which the form data should
46

```

```

JS userRoutes.js ×
routes > JS userRoutes.js > ...
1 const express = require('express');
2 const router = express.Router();
3
4 const authController = require('../controllers/authController');
5 router.get('/signup', authController.signup);
6 router.get('/login', authController.login);
7
8 router.post('/signup', authController.signup);
9 router.post('/login', authController.login);
10
11 module.exports = router;
12

```

Ln 6, Col 27 Spaces: 4 UTF-8 CRLF () JavaScript ✓ Prettier □

- **Middleware:** Middleware functions are used to execute code, modify the request and response objects, and end the request-response cycle. This project uses middleware for various purposes: Authentication Middleware (authMiddleware) checks if the user is authenticated and authorized to access certain routes. It includes JWT verification to ensure that the user has a valid token. Middleware functions are used in routes to protect certain endpoints. For example, routes for adding or editing books include authentication and authorization middleware to ensure only logged-in users can access these functionalities.

```

const jwt = require('jsonwebtoken');
const User = require('../models/userModel');

const authenticate = async (req, res, next) => {
  try {
    const token = req.cookies.jwt;
    const decodedToken = jwt.verify(token, process.env.JWT_SECRET);
    const userId = decodedToken.id || decodedToken._id;
    const authenticatedUser = await User.findById(userId);

    if (!authenticatedUser) {
      res.render('./login/loginForm', { title: " You are redirected as you are not Authenticated", user: undefined}); // the root view folder is views
    } else {
      req.user = authenticatedUser;
      next();
    }
  } catch {
    // however, we need to redirect to the login page in this case as we have a UI implemented in production
    res.render('./login/loginForm', { title: " You are redirected as you are not Authenticated", user: undefined}); // the root view folder is views
  }
}

const authorize = async (req, res, next) => {
  const currentUser = req.user;
  const userRole = currentUser.role;
  if(userRole){
    next();
  } else {
    res.status(403).json({ message: 'Forbidden' });
  }
};

async function fetchUserId(userId) {
  try {
    const user = await User.findById(userId);
    if (user) {
      console.log('Found user:', user);
    } else {
      console.log('No user found with the provided ID.');
    }
  } catch (error) {
    console.error('Error fetching user:', error);
  }
}

module.exports = { authenticate, authorize };

```

API specification:

Feature	Method	URL End Point
Register / Sign up	POST	<a href="http://localhost:4000/users/signup">http://localhost:4000/users/signup</a>
Login	POST	<a href="http://localhost:4000/users/login">http://localhost:4000/users/login</a>
View All Books	GET	<a href="http://localhost:4000/books">http://localhost:4000/books</a>
Add Book Exchange Form	GET	<a href="http://localhost:4000/books/newBookForm">http://localhost:4000/books/newBookForm</a>

Add Book Exchange	POST	<a href="http://localhost:4000/books">http://localhost:4000/books</a>
Get book by ID	GET	<a href="http://localhost:4000/books/:id">http://localhost:4000/books/:id</a>
Update Book Exchange Form	GET	<a href="http://localhost:4000/books/edit/:id">http://localhost:4000/books/edit/:id</a>
Update Book Exchange	POST	<a href="http://localhost:4000/books/update/:id">http://localhost:4000/books/update/:id</a>
Delete Book Exchange	POST	<a href="http://localhost:4000/books/delete/:id">http://localhost:4000/books/delete/:id</a>

## 1. Register / Sign up:

### Request:

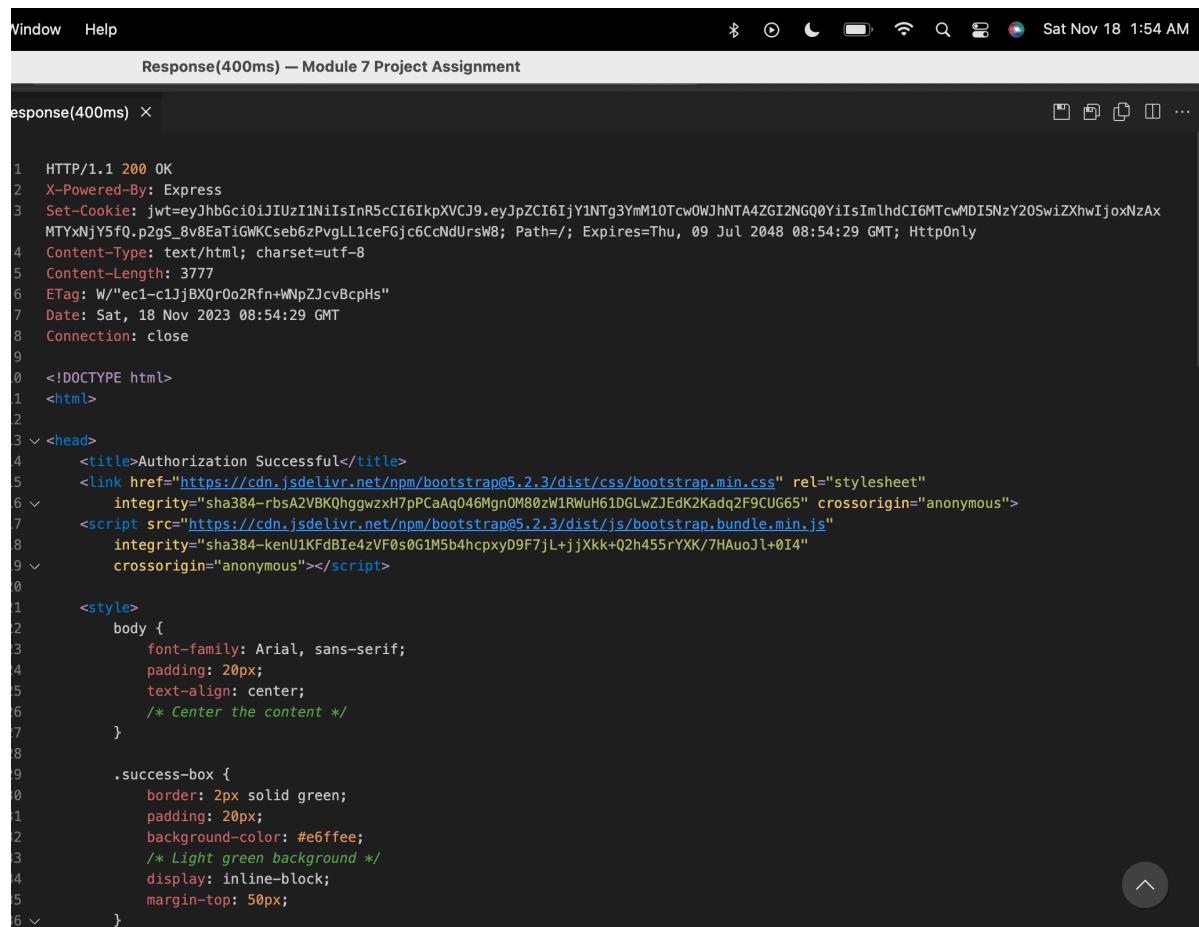
```
### Sample Test: register user
POST http://localhost:4000/users/signup
Content-Type: application/json

{
  "name": "Sushmita Prafull Halasawade",
  "email": "shalasaw@asu.edu",
  "password": "Password@1234",
  "passwordConfirmation": "Password@1234"
```

```
}
```

Upon a successful POST to '/signup', the server validates the data, creates a new user, issues a JWT as a secure cookie, and displays a success view with user details, sans the password. In case of errors, such as token issues or validation failures, it renders the login view with an appropriate error message.

## Response:



```
HTTP/1.1 200 OK
X-Powered-By: Express
Set-Cookie: jwt=eyJhbGciOiJIUzI1NiisInRcCI6IkpxVCJ9eyJpZCI6IjY1NTg3YmM1OTcw0WJhNTA4ZGI2NGQ0YiisImhdCI6MTcwMDI5NzY20SwiZXhwIjoxNzAxMTYxNjY5fQ.p2gS_8v8EaTiGWKCseb6zPvgLL1ceFGjc6CcNdUrsw8; Path=/; Expires=Thu, 09 Jul 2048 08:54:29 GMT; HttpOnly
Content-Type: text/html; charset=utf-8
Content-Length: 3777
ETag: W/"ec1-c1JjBXOrOo2Rfn+wNpZJcvBcpHs"
Date: Sat, 18 Nov 2023 08:54:29 GMT
Connection: close
<!DOCTYPE html>
<html>
<head>
    <title>Authorization Successful</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-sh3BgiKjnnn/dH+vqy+jte1T8PbW3+GzUcVwL6R8Nc4+HkQcGJZc+JLXz6JLjLcKq3sV" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBi4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+014" crossorigin="anonymous"></script>
<style>
    body {
        font-family: Arial, sans-serif;
        padding: 20px;
        text-align: center;
        /* Center the content */
    }
    .success-box {
        border: 2px solid green;
        padding: 20px;
        background-color: #e6ffee;
        /* Light green background */
        display: inline-block;
        margin-top: 50px;
    }
</style>
```

**Database: After user creation: user data is saved in users collection**

The screenshot shows the MongoDB Cloud Atlas interface. On the left, there's a sidebar with 'Project 0' selected under 'Data Services'. The main area shows the 'Cluster0' database with the 'Collections' tab selected. Under 'users', there's one document listed:

```

_id: ObjectId('65587bc59709ba508db64d4b')
name: "Sushmita Prafull Halasawade"
email: "shalasaw@asu.edu"
role: "client"
password: "$2a$12$QYWB0JU66xf0dqXnau/Eb0rrw.eSg3I8Nzn4XtXqsGThncRWsqgb$"
active: true
roles: ["client"]
__v: 0

```

## 2. Login user:

### Request:

```
### Sample Test: login user
```

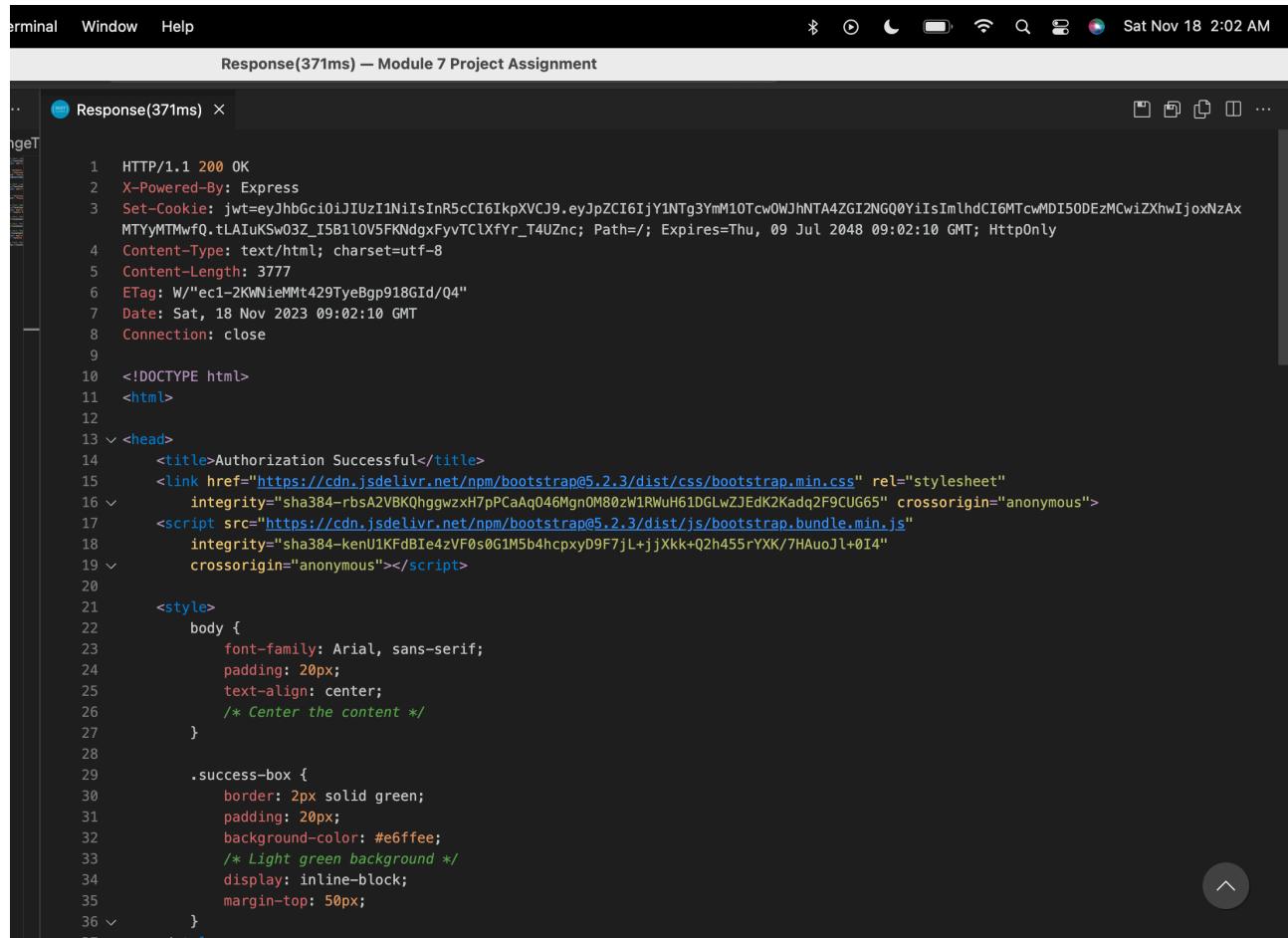
```
POST http://localhost:4000/users/login
```

```
Content-Type: application/json
```

```
{
  "email": "shalasaw@asu.edu",
  "password": "Password@1234"
}
```

"/login" endpoint in the Book Exchange website performs user authentication by verifying the provided email and password against the database. If authentication fails due to an incorrect email, password, or an error, it displays an error message on the login form; upon success, it generates and sends a token for further authentication.

## Response: jwt token



```
terminal Window Help
Response(371ms) — Module 7 Project Assignment
.. Response(371ms) ×
Sat Nov 18 2:02 AM
...
{
  "headers": [
    "HTTP/1.1 200 OK",
    "X-Powered-By: Express",
    "Set-Cookie: jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY1NTg3YmM10Tcw0WJhNTA4ZG12NGQ0YiIsImlhCI6MTcwMDI50DEzMCwiZXhwIjoxNzAxMTYyMTMwfQ.tLAiUKsw03Z_I5B1l0V5FKndgxFyvTClxYr_T4UZnc; Path=/; Expires=Thu, 09 Jul 2048 09:02:10 GMT; HttpOnly",
    "Content-Type: text/html; charset=utf-8",
    "Content-Length: 3777",
    "ETag: W/"ec1-2KwN1eMMt429TyeBgp918GId/Q4",
    "Date: Sat, 18 Nov 2023 09:02:10 GMT",
    "Connection: close"
  ],
  "body": "<!DOCTYPE html>\n<html>\n<head>\n  <title>Authorization Successful</title>\n  <link href=\"https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css\" rel=\"stylesheet\"\n        integrity=\"sha384-rbsA2VBKQhggwzxH7pPCaAq046Mgn0MB0zW1RWuH61DGLwZJEdK2Kadq2F9CUG65\" crossorigin=\"anonymous\"\n  <script src=\"https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js\"\n         integrity=\"sha384-kenU1KFdBe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4\"\n         crossorigin=\"anonymous\"></script>\n<style>\n  body {\n    font-family: Arial, sans-serif;\n    padding: 20px;\n    text-align: center;\n    /* Center the content */\n  }\n\n  .success-box {\n    border: 2px solid green;\n    padding: 20px;\n    background-color: #e6ffeef;\n    /* Light green background */\n    display: inline-block;\n    margin-top: 50px;\n  }\n</style>\n</head>\n<body>\n  <div class=\"success-box\" style=\"text-align: center; margin-top: 50px;\">\n    Authorization Successful!\n  </div>\n</body>\n</html>"
}
```

### 3. Add book:

#### Request:

```
### Sample Test: Add a new book
```

```
POST http://localhost:4000/books
```

```
Content-Type: application/json
```

```
Cookie:
```

```
jwt={eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY1NTg3YmM1OTcwOWJhNTA4  
ZGI2NGQ0YiIsImlhCI6MTcwMDMzODY2MCwiZXhwIjoxNzAxMjAyNjYwfQ.WceXRkym  
I058J-f1BwcsP8YQCd5wOxAoZlTLF5j4zGA}
```

```
{ "title": "The C Programming Language",  
  "author": "Brian W. Kernighan and Dennis M. Ritchie",  
  "description": "Regarded as the definitive guide to the C programming language, offering  
    concise and practical insights into programming with C",  
  "exchangeType": "borrow",  
  "owner": "65587bc59709ba508db64d4b",  
  "status": "available" }
```

the website creates a new Book object using request body data, including title, author, and owner, with access restricted by JWT-based authentication middleware. On successful book creation, it redirects to '/books', or displays an error page for any validation issues encountered.

**Response:** book created and redirected to /books

```

1 HTTP/1.1 302 Found
2 X-Powered-By: Express
3 Location: /books
4 Vary: Accept
5 Content-Type: text/plain; charset=utf-8
6 Content-Length: 28
7 Date: Sat, 18 Nov 2023 09:20:51 GMT
8 Connection: close
9
10 Found. Redirecting to /books

```

## Database: After addition of books:

The screenshot shows the MongoDB Cloud Atlas interface. On the left, there's a sidebar with navigation links like Project 0, Deployment, Services, Security, and Goto. The main area is titled "users.bookexchanges". It shows two documents in the "QUERY RESULTS: 1-5 OF 5" section:

```

_id: ObjectId('655881f39709ba508db64d51')
title: "The C Programming Language"
author: "Brian W. Kernighan and Dennis M. Ritchie"
description: "Regarded as the definitive guide to the C programming language, offeri..."
exchangeType: "borrow"
owner: ObjectId('65587bc59709ba508db64d4b')
status: "available"
createdAt: 2023-11-18T09:20:51.801+00:00
updatedAt: 2023-11-18T09:20:51.801+00:00
__v: 0

_id: ObjectId('655882c9709ba508db64d54')
title: "Programming Language Pragmatics"
author: "Michael L. Scott"
description: "Explores the design and implementation of programming languages, empha..."
exchangeType: "borrow"
owner: ObjectId('65587bc59709ba508db64d4b')
status: "available"
createdAt: 2023-11-18T09:20:51.801+00:00
updatedAt: 2023-11-18T09:20:51.801+00:00
__v: 0

```

## 4. Retrieve all books:

### Request:

```
GET http://localhost:4000/books
```

Cookie:

```
jwt={eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY1NTg3YmM1OTcwOWJhNTA4  
ZGI2NGQ0YiIsImhdCI6MTcwMDMzODY2MCwiZXhwIjoxNzAxMjAyNjYwfQ.WceXRkym  
I058J-f1BwcsP8YQCd5wOxAoZITLF5j4zGA}
```

The GET request route /books uses authentication and authorization middleware to verify JWT tokens in cookies, then calls getBooks, an asynchronous function that fetches all books from the database using Book.find(). If successful, it renders bookListForm with 'All Books', user info, and book data; if there's an error, it returns a 500 status and displays an appError view with the error message.

**Response:** 'bookListForm'

```
Window Help
BookExchangeTest.http — Module 7 Project Assignment
Sat Nov 18 2:22 AM
Response(174ms) ×
111 <thead>
112   <tr>
113     <th>Title</th>
114     <th>Author</th>
115     <th>Description</th>
116     <th>Exchange Type</th>
117     <th>Status</th>
118     <th>Actions</th>
119   </tr>
120 </thead>
121 <tbody>
122   <tr>
123     <td>
124       The C Programming Language
125     </td>
126     <td>
127       Brian W. Kernighan and Dennis M. Ritchie
128     </td>
129     <td>
130       Regarded as the definitive guide to the C programming language, offering concise and practical insights i
131       nto programming with C
132     </td>
133     <td>
134       borrow
135     </td>
136     <td>
137       available
138     </td>
139     <td>
140       <a href="/books/edit/655881f39709ba508db64d51">Edit</a> | <!-- Edit link -->
141       <form action="/books/delete/655881f39709ba508db64d51" method="post">
142         <input type="hidden" name="_method" value="delete">
143         <button type="submit" class="btn btn-danger">Delete</button>
144       </form>
145     </td>
146   </tr>
147 </tbody>
```

## Database:

The screenshot shows the MongoDB Cloud Atlas interface. On the left, there's a sidebar with sections like Deployment, Services, Security, and Advanced. The main area shows a database named 'users' with a collection named 'bookexchanges'. A search bar at the top says 'Search Namespaces'. Below it, there's a table with columns for '\_id', 'title', 'author', 'description', 'exchangeType', 'owner', 'status', 'createdAt', and 'updatedAt'. Two documents are listed:

```
_id: ObjectId('655881f39709ba508db64d51')
title: "The C Programming Language"
author: "Brian W. Kernighan and Dennis M. Ritchie"
description: "Regarded as the definitive guide to the C programming language, offers...""
exchangeType: "borrow"
owner: ObjectId('655887bc59709ba508db64d4b')
status: "available"
createdAt: 2023-11-18T09:20:51.801+00:00
updatedAt: 2023-11-18T09:20:51.801+00:00
__v: 0

_id: ObjectId('6558822c9709ba508db64d54')
title: "Programming Language Pragmatics"
author: "Michael L. Scott"
description: "Explores the design and implementation of programming languages, empha...""
exchangeType: "borrow"
owner: ObjectId('655887bc59709ba508db64d4b')
__v: 0
```

## 5. Get book by ID:

### Request:

```
### Sample Test: Fetch book by ID
```

```
GET http://localhost:4000/books/6558822e9709ba508db64d57
```

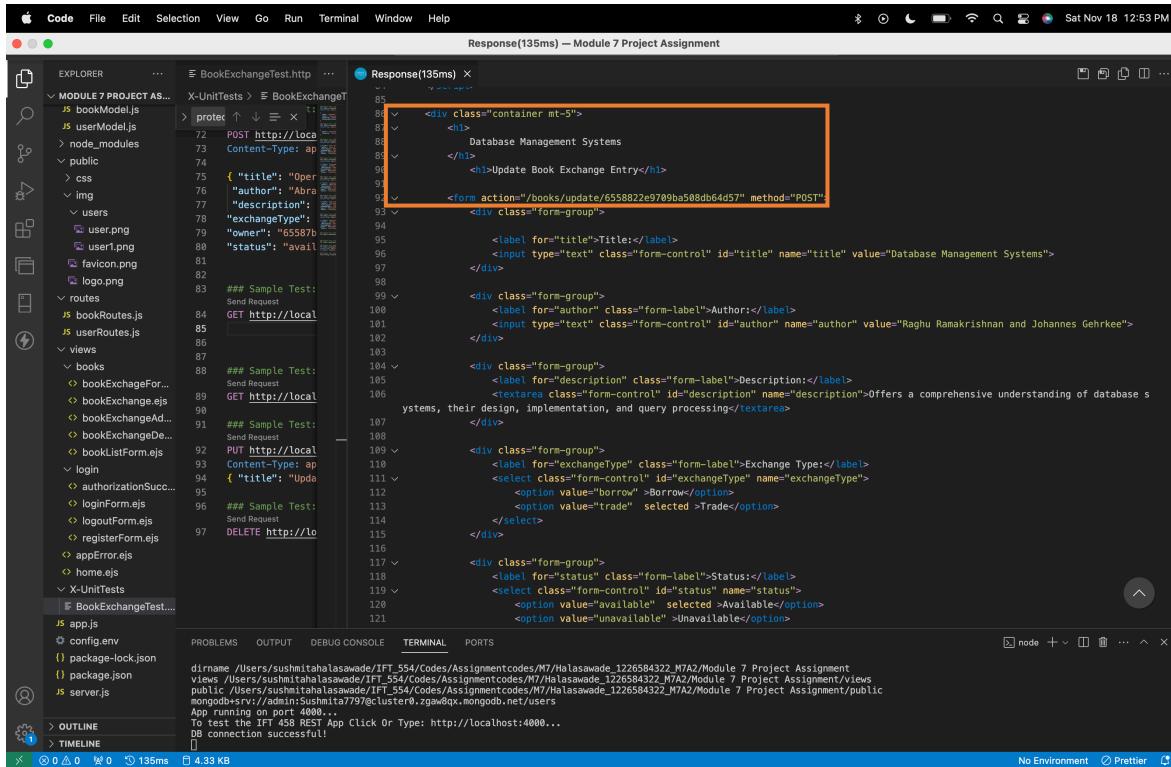
Cookie:

```
jwt={eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY1NTg3YmM1OTcwOWJhNTA4
ZGI2NGQ0YiIsImlhCI6MTcwMDMzODY2MCwiZXhwIjoxNzAxMjAyNjYwfQ.WceXRkym
I058J-f1BwcsP8YQCd5wOxAoZlTLF5j4zGA}
```

The getBookById function asynchronously retrieves a book from the database using its ID, secured by authentication and authorization middleware using JWT in cookies. It renders book

exchange details if found, or returns a 404 error for missing books, and a 500 status for other errors.

## Response: book exchange details



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

- File Path:** BookExchangeTest.http
- Content:**

```

POST http://localhost:4000/books/update/6558822e9709ba508db64d57" method="POST"
      
```
- Code Editor Area:**

```

<div class="container mt-5">
  <h1>Database Management Systems</h1>
  <h2>Update Book Exchange Entry</h2>
  <form action="/books/update/6558822e9709ba508db64d57" method="POST">
    <div class="form-group">
      <label for="title">Title:</label>
      <input type="text" class="form-control" id="title" name="title" value="Database Management Systems">
    </div>
    <div class="form-group">
      <label for="author">Author:</label>
      <input type="text" class="form-control" id="author" name="author" value="Raghuram Krishnan and Johannes Gehrke">
    </div>
    <div class="form-group">
      <label for="description">Description:</label>
      <textarea class="form-control" id="description" name="description">Offers a comprehensive understanding of database systems, their design, implementation, and query processing</textarea>
    </div>
    <div class="form-group">
      <label for="exchangeType">Exchange Type:</label>
      <select class="form-control" id="exchangeType" name="exchangeType">
        <option value="borrow">Borrow</option>
        <option value="trade" selected>Trade</option>
      </select>
    </div>
    <div class="form-group">
      <label for="status">Status:</label>
      <select class="form-control" id="status" name="status">
        <option value="available" selected>Available</option>
        <option value="unavailable">Unavailable</option>
      </select>
    </div>
  </form>

```
- Terminal Output:**

```

$ node app.js
[{"id": "6558822e9709ba508db64d57", "title": "Database Management Systems", "author": "Raghuram Krishnan and Johannes Gehrke", "description": "Offers a comprehensive understanding of database systems, their design, implementation, and query processing", "exchangeType": "trade", "status": "available"}]

```

## Database:

The screenshot shows the MongoDB Cloud Atlas interface. On the left, there's a sidebar with sections like Deployment, Services, and Security. The main area is titled 'users.bookexchanges' and shows document details. One document is highlighted with an orange border:

```

_id: ObjectId("6558822e9709ba508db64d57")
title: "Database Management Systems"
author: "Raghuramakrishnan and Johannes Gehrke"
description: "Offers a comprehensive understanding of database systems, their design, and management."
exchangeType: "trade"
owner: ObjectId("65587bc59709ba508db64d4b")
status: "available"
createdAt: 2023-11-18T09:21:50.271+00:00
updatedAt: 2023-11-18T09:21:50.271+00:00
__v: 0

```

## 6. Update book:

**Request:**

### Sample Test: Modify a book by ID

POST http://localhost:4000/books/update/6558822e9709ba508db64d57

Content-Type: application/json

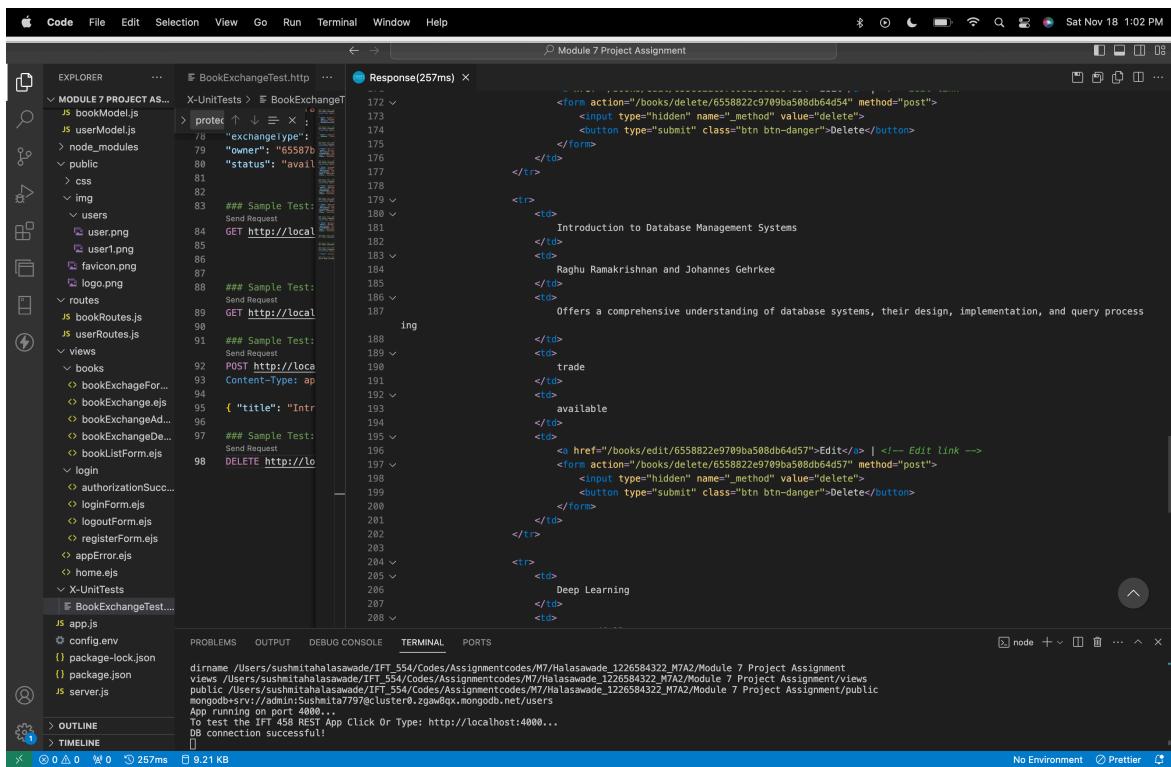
Cookie:

jwt={eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY1NTg3YmM1OTcwOWJhNTA4ZGI2NGQ0YiIsImlhCI6MTcwMDMzODY2MCwiZXhwIjoxNzAxMjAyNjYwfQ.WceXRkymI058J-f1BwesP8YQCd5wOxAoZlTLF5j4zGA}

{ "title": "Introduction to Database Management Systems" }

The updateBookById function in the Book Exchange website updates a book using its ID and req.body data with the findByIdAndUpdate method. On success, it shows the updated book list, while failures lead to a 404 error page or a 500 status with an error message in case of other errors.

## Response: book list



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

- File Structure:** The left sidebar shows a project structure for "Module 7 Project Assignment" with files like bookModel.js, userModel.js, routes/bookRoutes.js, routes/userRoutes.js, views/books, views/auth, and test files like BookExchangeTest.js and X-UnitTests.js.
- Terminal Tab:** The right side has a "Response(257ms)" tab showing the HTML content of a book list page. The content includes a table with columns for title, author, and availability, and links for edit and delete actions.
- Output Tab:** Below the terminal, the "OUTPUT" tab displays logs from the application's console, including the path to the assignment code and a MongoDB connection message.
- Bottom Status Bar:** Shows the file count (0), line count (0), character count (257ms), and file size (9.21 KB).

```

<form action="/books/delete/6558822e9789ba588db64d57" method="post">
  <input type="hidden" name="_method" value="delete">
  <button type="submit" class="btn btn-danger">Delete</button>
</form>
</td>
</tr>
<td>
  Introduction to Database Management Systems
</td>
<td>
  Raghu Ramakrishnan and Johannes Gehrke
</td>
<td>
  Offers a comprehensive understanding of database systems, their design, implementation, and query process
</td>
<td>
  trade
</td>
<td>
  available
</td>
<td>
    <a href="/books/edit/6558822e9789ba588db64d57">Edit</a> | <!-- Edit link -->
    <form action="/books/delete/6558822e9789ba588db64d57" method="post">
      <input type="hidden" name="_method" value="delete">
      <button type="submit" class="btn btn-danger">Delete</button>
    </form>
</td>
<tr>
<td>
  Deep Learning
</td>

```

Database:Before updating database management book:

The screenshot shows the MongoDB Cloud Atlas interface. On the left, a sidebar navigation includes 'Project 0', 'Data Services', 'App Services', and 'Charts'. Under 'Data Services', 'Database' is selected, showing databases like 'IFT-554-2023', 'LoanManagement', 'Student\_Course', 'demodb', 'ift\_554\_DB', 'mydatabase', 'user\_col', and 'users'. Under 'users', 'bookexchanges' is selected. The main panel displays the 'bookexchanges' collection with 13 documents. A specific document is highlighted with an orange border:

```

_id: ObjectId('6558822e9709ba508db64d57')
title: "Database Management Systems"
author: "Raghuramakrishnan and Johannes Gehrke"
description: "Offers a comprehensive understanding of database systems, their design, and management."
exchangeType: "trade"
owner: ObjectId('65587bc59709ba508db64d4b')
status: "available"
createdAt: 2023-11-18T09:21:50.271+00:00
updatedAt: 2023-11-18T09:21:50.271+00:00
_v: 0

```

After updating the title intro to database management:

The screenshot shows the MongoDB Cloud Atlas interface after an update. The 'bookexchanges' collection now contains a single document with the following updated title:

```

_id: ObjectId('6558822e9709ba508db64d5a')
title: "Introduction to Database Management Systems"
author: "Raghuramakrishnan and Johannes Gehrke"
description: "Offers a comprehensive understanding of database systems, their design, and management."
exchangeType: "trade"
owner: ObjectId('65587bc59709ba508db64d4b')
status: "available"
createdAt: 2023-11-18T09:21:50.271+00:00
updatedAt: 2023-11-18T09:21:50.271+00:00
_v: 0

```

## 7. Delete book:

### Request:

```
### Sample Test: Delete book by ID
```

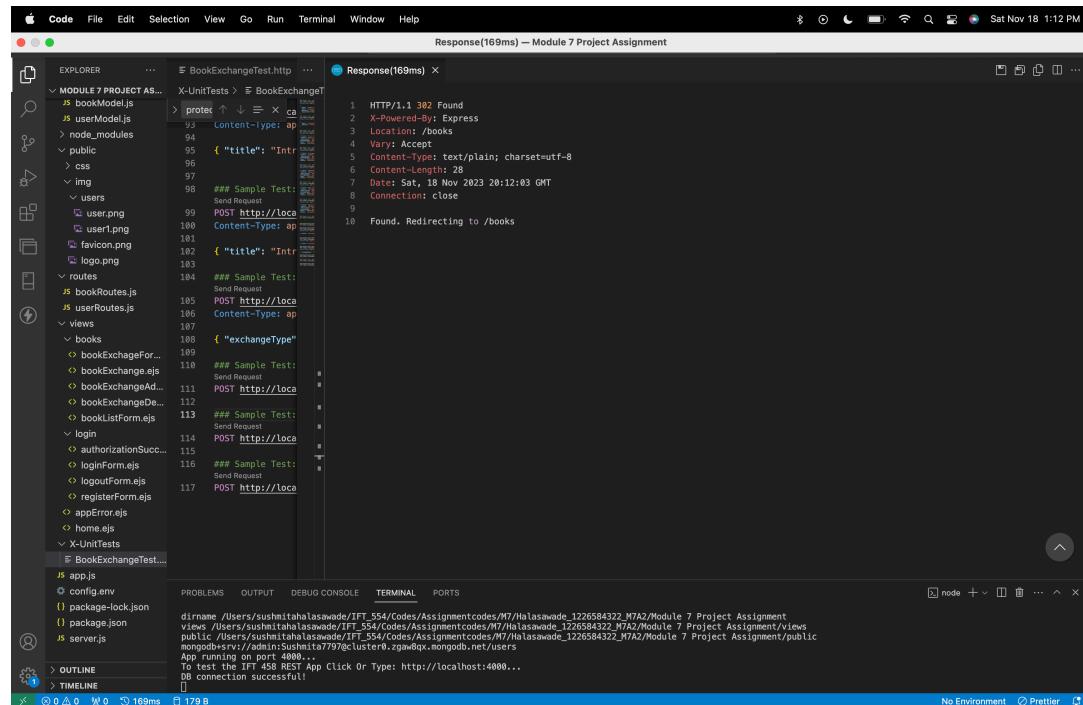
```
POST http://localhost:4000/books/delete/6558822e9709ba508db64d57
```

Cookie:

```
jwt={eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY1NTg3YmM1OTcwOWJhNTA4  
ZGI2NGQ0YiIsImlhCI6MTcwMDMzODY2MCwiZXhwIjoxNzAxMjAyNjYwfQ.WceXRkym  
I058J-f1BwcsP8YQCd5wOxAoZlTLF5j4zGA}
```

The deleteBook function uses `findByIdAndRemove` to delete the book by its ID, redirecting to `'/books'` upon success, or rendering a 404 error if not found. Protected by authentication and authorization middleware, it validates user access via a JWT token stored in a cookie.

### Response:



```
HTTP/1.1 302 Found
X-Powered-By: Express
Location: /books
Content-Type: text/plain; charset=utf-8
Content-Length: 28
Date: Sat, 18 Nov 2023 20:12:03 GMT
Connection: close

Found, Redirecting to /books
```

The screenshot shows a terminal window with the following details:

- File Path:** BookExchangeTest.http
- Line 1:** HTTP/1.1 302 Found
- Line 2:** X-Powered-By: Express
- Line 3:** Location: /books
- Line 4:** Content-Type: text/plain; charset=utf-8
- Line 5:** Content-Length: 28
- Line 6:** Date: Sat, 18 Nov 2023 20:12:03 GMT
- Line 7:** Connection: close
- Line 8:** Found, Redirecting to /books

Below the terminal window, the file structure of the project is visible, including files like `bookModel.js`, `userModel.js`, `node_modules`, `public`, `css`, `img`, `users`, `routes`, and `views`.

## After deleting books:

The screenshot shows the MongoDB Cloud interface. On the left, the sidebar lists databases and collections. Under the 'Database' section, the 'users' database is selected, and its 'bookexchanges' collection is shown. A search bar at the top right contains the query: { field: 'value' }. Below the search bar, there are two documents listed:

```
_id: ObjectId('655881f39709ba508db64d51')
title: "The C Programming Language"
author: "Brian W. Kernighan and Dennis M. Ritchie"
description: "Regarded as the definitive guide to the C programming language, offering clear explanations of the language's features and how to use them effectively."
exchangeType: "borrow"
owner: ObjectId('65587bc59709ba508db64d4b')
status: "available"
createdAt: 2023-11-18T09:20:51.801+00:00
updatedAt: 2023-11-18T09:20:51.801+00:00
__v: 0
```

```
_id: ObjectId('6558822c9709ba508db64d54')
title: "Programming Language Pragmatics"
author: "Michael L. Scott"
description: "Explores the design and implementation of programming languages, emphasizing practical aspects and real-world examples."
exchangeType: "borrow"
owner: ObjectId('65587bc59709ba508db64d4b')
```

The status of the second document is red, indicating it has been deleted.

## Screenshots:

### Home page:

The screenshot shows the Book Exchange Platform home page. At the top, the browser menu includes Chrome, File, Edit, View, History, Bookmarks, Profiles, Tab, Window, Help, and a Relaunch to update button. The address bar shows the URL <http://localhost:4000>. The main content area displays "Batch of IFT 458 Fall 2023" with a graduation cap icon. It shows the server time as 11/18/2023, 1:06:20 AM, and the IP address as 75.233.80.26. A "Student Information" section includes a student image, name (Sushmita Prafull Halasawade), email (shalasaw@asu.edu), ID (1226584322), and course (#IFT554). Below this is a "Welcome to Book Exchange!" section with "Book Exchange", "Register", and "Login" buttons. At the bottom, it says "Available Books:" and "Book Exchange Platform © IFT 510 - Fall 2023".

### Register:

The screenshot shows the Book Exchange Platform registration page. The browser menu and address bar are identical to the home page. The main content area displays "Batch of IFT 458 Fall 2023" with a graduation cap icon. It shows the server time as 11/18/2023, 1:06:43 AM, and the IP address as 75.233.80.26. A "Student Information" section includes a student image, name (Sushmita Prafull Halasawade), email (shalasaw@asu.edu), ID (1226584322), and course (#IFT554). Below this is a "Register >" section with input fields for Name (Sushmita Prafull Halasawade), Email (shalasaw@asu.edu), Password (\*\*\*\*\*), Confirm Password (\*\*\*\*\*), and a "Signup" button. A link "Already have an account? [Login](#)" is also present.

## Auth success:

A screenshot of a Chrome browser window. The title bar says "Authorization Successful". The address bar shows "http://localhost:4000/users/signup". The main content area displays "Batch of IFT 458 Fall 2023" with a graduation cap icon. It shows "Server: 11/18/2023, 1:07:17 AM" and "Your IP address is: 75.233.80.26". A "Student Information" section includes a student image, name (Sushmita Prafull Halasawade), email (shalasaw@asu.edu), ID (1226584322), and course (#IFT554). A large green box titled "Authorization Successful!" contains a welcome message, email, token (a long string of characters), and a note: "You can now access the platform's features."

## Login:

A screenshot of a Chrome browser window. The title bar says "Login". The address bar shows "http://localhost:4000/users/login". The main content area displays "Batch of IFT 458 Fall 2023" with a graduation cap icon. It shows "Server: 11/18/2023, 1:07:43 AM" and "Your IP address is: 75.233.80.26". A "Student Information" section includes a student image, name (Sushmita Prafull Halasawade), email (shalasaw@asu.edu), ID (1226584322), and course (#IFT554). Below this is a "Login" form with fields for "email" (shalasaw@asu.edu) and "Password" (redacted). A "Login" button and a link "Don't have an account? [Sign-up](#)" are at the bottom.

## Book exchange:

The screenshot shows a Chrome browser window with the URL <http://localhost:4000/books>. The page title is "Batch of IFT 458 Fall 2023". It displays student information for Sushmita Prafull Halasawade, including a profile picture, name, email, ID, and course details. Below this is a "Book List" section with a "Add New Book" button and a table header row with columns: Title, Author, Description, Exchange Type, Status, Actions.

## Add new book:

The screenshot shows a Chrome browser window with the URL <http://localhost:4000/books/newBookForm>. The page title is "Book Exchange Platform - Add". It displays student information for Sushmita Prafull Halasawade. Below this is an "Add New Book" form with fields for Title, Author, Description, Exchange Type, and Status. The "Title" field contains "Intro to database", "Author" field contains "Sushmita", "Description" field contains "test book", "Exchange Type" field contains "Borrow", and "Status" field contains "Available". A blue "Add Book" button is at the bottom.

## Update book:

Student Name: Sushmita Prafull Halasawade  
Student Email: shalasaw@asu.edu  
Student ID: 1226584322  
Course #: IFT554

### Intro to database

### Update Book Exchange Entry

Title: Intro to database

Author: Sushmita

Description: test book

Exchange Type: Borrow

Status: Available

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

Batch of IFT 458 Fall 2023 

Server: 11/18/2023, 1:09:37 AM  
Your IP address is: 75.233.80.26

#### Student Information

Student Image: 

Student Name: Sushmita Prafull Halasawade  
Student Email: shalasaw@asu.edu  
Student ID: 1226584322  
Course #: IFT554

#### Book List

Add New Book

Title	Author	Description	Exchange Type	Status	Actions
Intro to database	Sushmita Halasawade	test book	borrow	available	<a href="#">Edit</a>   <a href="#">Delete</a>

[Go to home](#)

## delete book:

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

- Address Bar:** localhost:4000/books
- Page Content:**
  - Batch of IFT 458 Fall 2023:** Includes a graduation cap icon.
  - Server:** 11/18/2023, 1:09:53 AM
  - Your IP address is:** 75.233.80.26
  - Student Information:**
    - Student Image:** A small thumbnail of a person with long dark hair.
    - Student Name:** Sushmita Prafull Halasawade
    - Student Email:** shalasaw@asu.edu
    - Student ID:** 1226584322
    - Course #:** IFT554
  - Book List:** A section titled "Book List" with a "Add New Book" button. Below it is a table header row with columns: Title, Author, Description, Exchange Type, Status, Actions.
  - Links:** "Go to home"

## Adding 5 books in the bookexchange:

Before addition: empty book list

Batch of IFT 458 Fall 2023 

Server:  
11/18/2023, 1:09:53 AM  
Your IP address is: 75.233.80.26

**Student Information**

Student Image: 

Student Name: Sushmita Prafull Halasawade  
Student Email: shalasew@asu.edu  
Student ID: 1226584322  
Course #: IFT554

**Book List**

Add New Book

Title	Author	Description	Exchange	Type	Status	Actions
<a href="#">Go to home</a>						

After addition of 5 books:

Batch of IFT 458 Fall 2023 

Server:  
11/18/2023, 1:26:28 AM  
Your IP address is: 75.233.80.26

**Student Information**

Student Image: 

Student Name: Sushmita Prafull Halasawade  
Student Email: shalasew@asu.edu  
Student ID: 1226584322  
Course #: IFT554

**Book List**

Add New Book

Title	Author	Description	Exchange	Type	Status	Actions
To Kill a Mockingbird	Harper Lee	A classic novel addressing racial injustice and moral growth in a small Southern town through the eyes of a young girl. borrow	available			<a href="#">Edit</a>   <a href="#">Delete</a>
The Lord of the Rings	J.R.R. Tolkien	An epic fantasy trilogy chronicling the quest to destroy the One Ring and save Middle-earth from the dark lord Sauron. trade	available			<a href="#">Edit</a>   <a href="#">Delete</a>
The Alchemist	Paulo Coelho	A philosophical novel about following dreams and understanding the language of the universe	trade		available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Great Gatsby	F. Scott Fitzgerald	An exploration of the American Dream and its disillusionment during the Roaring Twenties.	borrow		available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Harry Potter series	J.K. Rowling	Chronicles the journey of a young wizard, Harry Potter, and his battle against the dark wizard Voldemort.	trade		available	<a href="#">Edit</a>   <a href="#">Delete</a>
<a href="#">Go to home</a>						

Database: 5 documents added in book exchange collection.

**users.bookexchanges**

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 136KB TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

**Find** Indexes Schema Anti-Patterns Aggregation Search Indexes

**INSERT DOCUMENT**

**QUERY RESULTS: 1-5 OF 5**

```
_id: ObjectId('6558729e361f86db9d77235f')
title: "To Kill a Mockingbird"
author: "Harper Lee"
description: "A classic novel addressing racial injustice and moral growth in a small Southern town through the eyes of a young girl. borrow"
exchangeType: "borrow"
owner: ObjectId('655870b5361f86db9d77233d')
status: "available"
createdAt: 2023-11-18T08:15:26.597+00:00
updatedAt: 2023-11-18T08:18:50.370+00:00
__v: 0

_id: ObjectId('655872e7361f86db9d772374')
title: "The Lord of the Rings"
author: "J.R.R. Tolkien"
description: "An epic fantasy trilogy chronicling the quest to destroy the One Ring -"
exchangeType: "trade"
owner: ObjectId('655870b5361f86db9d77233d')
```

System Status: All Good

©2023 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Before updating book: to kill a mockingbird:

Batch of IFT 458 Fall 2023

Server: 11/18/2023, 1:26:26 AM  
Your IP address is: 75.233.80.26

**Student Information**

Student Image:

Student Name: Sushmita Prafull Halasawade  
Student Email: shalasaw@asu.edu  
Student ID: 1226584322  
Course #: IFT554

**Book List**

Add New Book

Title	Author	Description	Exchange Type	Status	Actions
To Kill a Mockingbird	Harper Lee	A classic novel addressing racial injustice and moral growth in a small Southern town through the eyes of a young girl. borrow	borrow	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Lord of the Rings	J.R.R. Tolkien	An epic fantasy trilogy chronicling the quest to destroy the One Ring and save Middle-earth from the dark lord Sauron. trade	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Alchemist	Paulo Coelho	A philosophical novel about following dreams and understanding the language of the universe	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Great Gatsby	F. Scott Fitzgerald	An exploration of the American Dream and its disillusionment during the Roaring Twenties.	borrow	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Harry Potter series	J.K. Rowling	Chronicles the journey of a young wizard, Harry Potter, and his battle against the dark wizard Voldemort.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>

[Go to home](#)

After updating the book: exchange changed from borrow to trade.

The screenshot shows a Chrome browser window with the URL <http://localhost:4000/books/update/6558729e361f86db9d77235f>. The page displays a student profile and a table of books.

**Student Information:**

- Batch of IFT 458 Fall 2023
- Server: 11/18/2023, 1:28:02 AM
- Your IP address is: 75.233.80.26
- Student Name: Sushmita Prafull Halasawade
- Student Email: shalasaw@asu.edu
- Student ID: 1226584322
- Course #: IFT554

**Book List**

Add New Book

Title	Author	Description	Exchange Type	Status	Actions
To Kill a Mockingbird	Harper Lee	A classic novel addressing racial injustice and moral growth in a small Southern town through the eyes of a young girl.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Lord of the Rings	J.R.R. Tolkien	An epic fantasy trilogy chronicling the quest to destroy the One Ring and save Middle-earth from the dark lord Sauron.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Alchemist	Paulo Coelho	A philosophical novel about following dreams and understanding the language of the universe	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Great Gatsby	F. Scott Fitzgerald	An exploration of the American Dream and its disillusionment during the Roaring Twenties.	borrow	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Harry Potter series	J.K. Rowling	Chronicles the journey of a young wizard, Harry Potter, and his battle against the dark wizard Voldemort.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>

[Go to home](#)

Database:

The screenshot shows the MongoDB Cloud Atlas interface. On the left, a sidebar lists 'Project 0' under 'Data Services'. The main area is titled 'ClusterO' and shows the 'users.bookexchanges' collection. It displays storage information: 34KB logical size, 1.5KB total document size, 5 documents, and 36KB index size. A query results table shows one document:

```

_id: ObjectId('6558729e361f86db9d77235f')
title: "To Kill a Mockingbird"
author: "Harper Lee"
description: "A classic novel addressing racial injustice and moral growth in a small Southern town through the eyes of a young girl."
exchangeType: "trade"
owner: ObjectId("655870b5361f86db9d77233d")
status: "available"
createdAt: 2023-11-18T08:15:26.597+00:00
updatedAt: 2023-11-18T08:28:02.029+00:00
...v: ...

```

Before updating book: the great Gatsby:

The screenshot shows a web application at <http://localhost:4000/books/update>. It displays a 'Student Information' section with a student's profile picture and details: Name: Sushmita Prafull Halasawade, Email: shalasaw@asu.edu, ID: 1226584322, Course: IFT554. Below this is a 'Book List' section with a table:

Title	Author	Description	Exchange Type	Status	Actions
To Kill a Mockingbird	Harper Lee	A classic novel addressing racial injustice and moral growth in a small Southern town through the eyes of a young girl.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Lord of the Rings	J.R.R. Tolkien	An epic fantasy trilogy chronicling the quest to destroy the One Ring and save Middle-earth from the dark lord Sauron.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Alchemist	Paulo Coelho	A philosophical novel about following dreams and understanding the language of the universe	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Great Gatsby	F. Scott Fitzgerald	An exploration of the American Dream and its disillusionment during the Roaring Twenties.	borrow	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Harry Potter series	J.K. Rowling	Chronicles the journey of a young wizard, Harry Potter, and his battle against the dark wizard Voldemort.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>

[Go to home](#)

After updation of description:

The screenshot shows a Chrome browser window with the URL <http://localhost:4000/books/update/>. The page displays a student profile and a table of books.

**Student Information:**

- Batch of IFT 458 Fall 2023
- Server: 11/18/2023, 1:29:34 AM
- Your IP address is: 75.233.80.26
- Student Information:  
Student Name: Sushmita Prafull Halasawade  
Student Email: shalasaw@asu.edu  
Student ID: 1226584322  
Course #: IFT554

**Book List**

Add New Book

Title	Author	Description	Exchange	Type	Status	Actions
To Kill a Mockingbird	Harper Lee	A classic novel addressing racial injustice and moral growth in a small Southern town through the eyes of a young girl.	trade		available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Lord of the Rings	J.R.R. Tolkien	An epic fantasy trilogy chronicling the quest to destroy the One Ring and save Middle-earth from the dark lord Sauron.	trade		available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Alchemist	Paulo Coelho	A philosophical novel about following dreams and understanding the language of the universe		trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Great Gatsby	F. Scott Fitzgerald	A captivating tale set in the Roaring Twenties, delving into the elusive American Dream and its consequences.	borrow		available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Harry Potter series	J.K. Rowling	Chronicles the journey of a young wizard, Harry Potter, and his battle against the dark wizard Voldemort.	trade		available	<a href="#">Edit</a>   <a href="#">Delete</a>

[Go to home](#)

Database:

The screenshot shows the MongoDB Cloud Atlas interface. On the left, there's a sidebar with sections for Overview, Deployment (Database, Services, Security), and Project 0. The main area is titled 'Collections' and shows the 'bookexchanges' collection. It lists 5 documents with their IDs and titles. One document is highlighted with an orange border, showing its full JSON structure:

```

{
  "_id": ObjectId('6558751b361f86db9d77239f'),
  "title": "The Great Gatsby",
  "author": "F. Scott Fitzgerald",
  "description": "A captivating tale set in the Roaring Twenties, delving into the elusive American Dream and its consequences.",
  "exchangeType": "borrow",
  "owner": ObjectId('6558705361f86db9d77233d'),
  "status": "available",
  "createdAt": 2023-11-18T08:26:03.063+00:00,
  "updatedAt": 2023-11-18T08:29:34.102+00:00
}

```

Before deleting:

The screenshot shows a web application for managing books. At the top, it says 'Batch of IFT 458 Fall 2023' with a graduation cap icon. Below that, it shows the date (11/18/2023, 1:29:34 AM) and IP address (75.233.80.26). The 'Student Information' section shows a student's profile picture and details: Name: Sushmita Prafull Halasawade, Email: shalasaw@asu.edu, ID: 1226584322, Course: IFT554.

## Book List

Add New Book

Title	Author	Description	Exchange Type	Status	Actions
To Kill a Mockingbird	Harper Lee	A classic novel addressing racial injustice and moral growth in a small Southern town through the eyes of a young girl.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Lord of the Rings	J.R.R. Tolkien	An epic fantasy trilogy chronicling the quest to destroy the One Ring and save Middle-earth from the dark lord Sauron.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Alchemist	Paulo Coelho	A philosophical novel about following dreams and understanding the language of the universe	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Great Gatsby	F. Scott Fitzgerald	A captivating tale set in the Roaring Twenties, delving into the elusive American Dream and its consequences.	borrow	available	<a href="#">Edit</a>   <a href="#">Delete</a>
The Harry Potter series	J.K. Rowling	Chronicles the journey of a young wizard, Harry Potter, and his battle against the dark wizard Voldemort.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>

[Go to home](#)

After deleting book great Gatsby and harry potter series:

The screenshot shows a Chrome browser window on a Mac OS X desktop. The title bar indicates it's Saturday, November 18, at 1:31 AM. The address bar shows the URL <http://localhost:4000/books>. The page content is as follows:

Batch of IFT 458 Fall 2023

Server:  
11/18/2023, 1:31:09 AM  
Your IP address is: 75.233.80.26

**Student Information**

Student Image:

Student Name: Sushmita Prafull Halasawade  
Student Email: shalasav@asu.edu  
Student ID: 1226554322  
Course #: IFT554

**Book List**

Add New Book

Title	Author	Description	Exchange	Type	Status	Actions
To Kill a Mockingbird	Harper Lee	A classic novel addressing racial injustice and moral growth in a small Southern town through the eyes of a young girl.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>	
The Lord of the Rings	J.R.R. Tolkien	An epic fantasy trilogy chronicling the quest to destroy the One Ring and save Middle-earth from the dark lord Sauron.	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>	
The Alchemist	Paulo Coelho	A philosophical novel about following dreams and understanding the language of the universe	trade	available	<a href="#">Edit</a>   <a href="#">Delete</a>	

[Go to home](#)

## Database: 3books available in book exchange

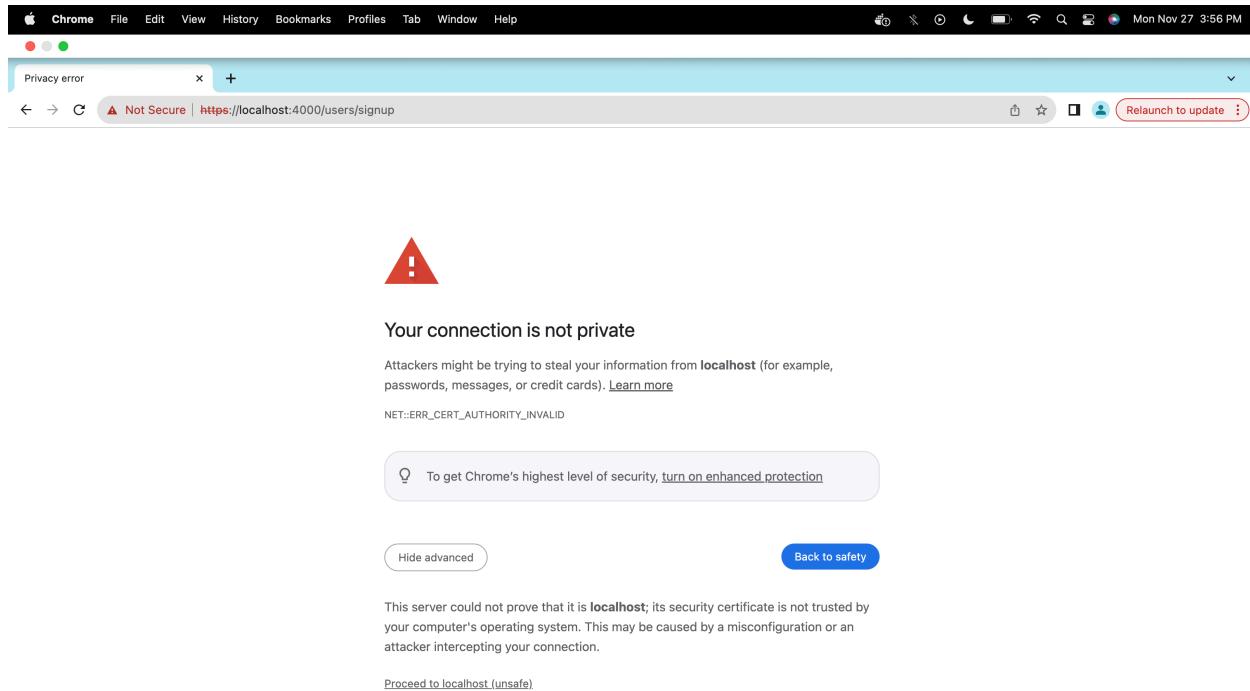
The screenshot shows the MongoDB Cloud Atlas interface. On the left, there's a sidebar with sections like Overview, Deployment, Database, Services, Security, and Goto. The main area is titled 'Cluster0' and shows the 'users' database. Under 'Collections', it lists 'bookexchanges' (which has 3 documents) and 'users'. A search bar at the top says 'Search Namespaces'. Below the collections, there's a 'Find' button and a query input field: 'Type a query: { field: 'value' }'. The results section is titled 'QUERY RESULTS: 1-3 OF 3' and displays the following document:

```
_id: ObjectId('6558729e361f86db9d77235f')
title: "To Kill a Mockingbird"
author: "Harper Lee"
description: "A classic novel addressing racial injustice and moral growth in a small town in the 1930s."
exchangeType: "trade"
owner: ObjectId('655870b5361f86db9d77233d')
status: "available"
createdAt: 2023-11-18T08:15:26.597+00:00
updatedAt: 2023-11-18T08:18:02.029+00:00
__v: 0

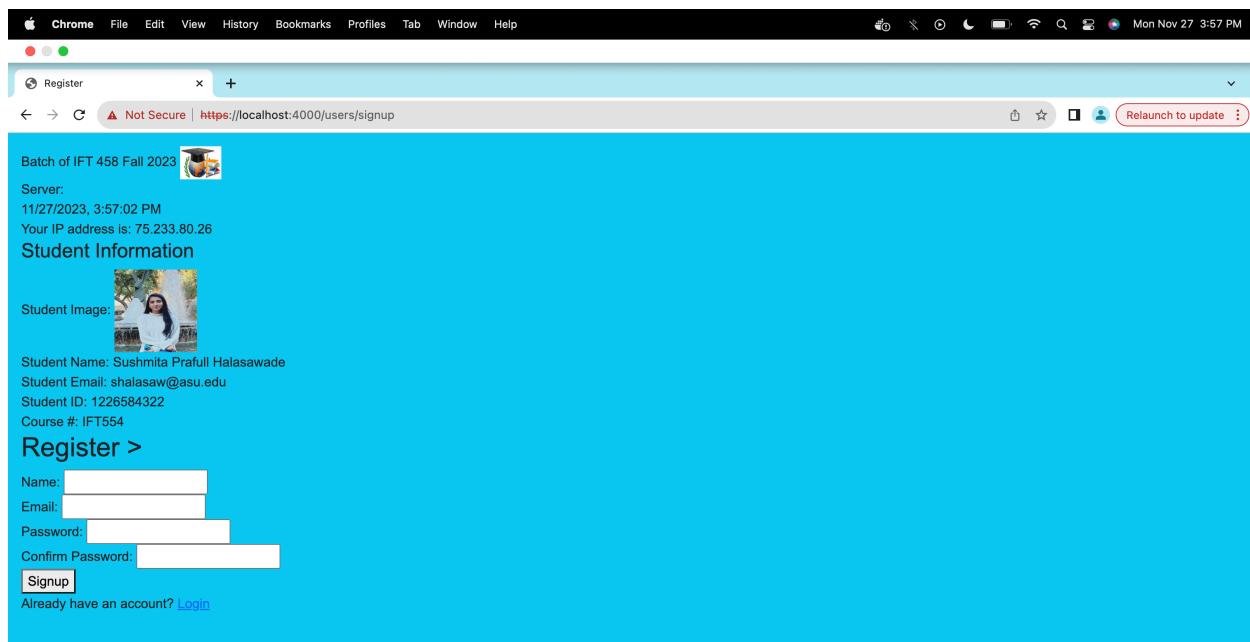
id: ObjectId('655872e7361f86db9d772374')
```

## HTTPS implementation:

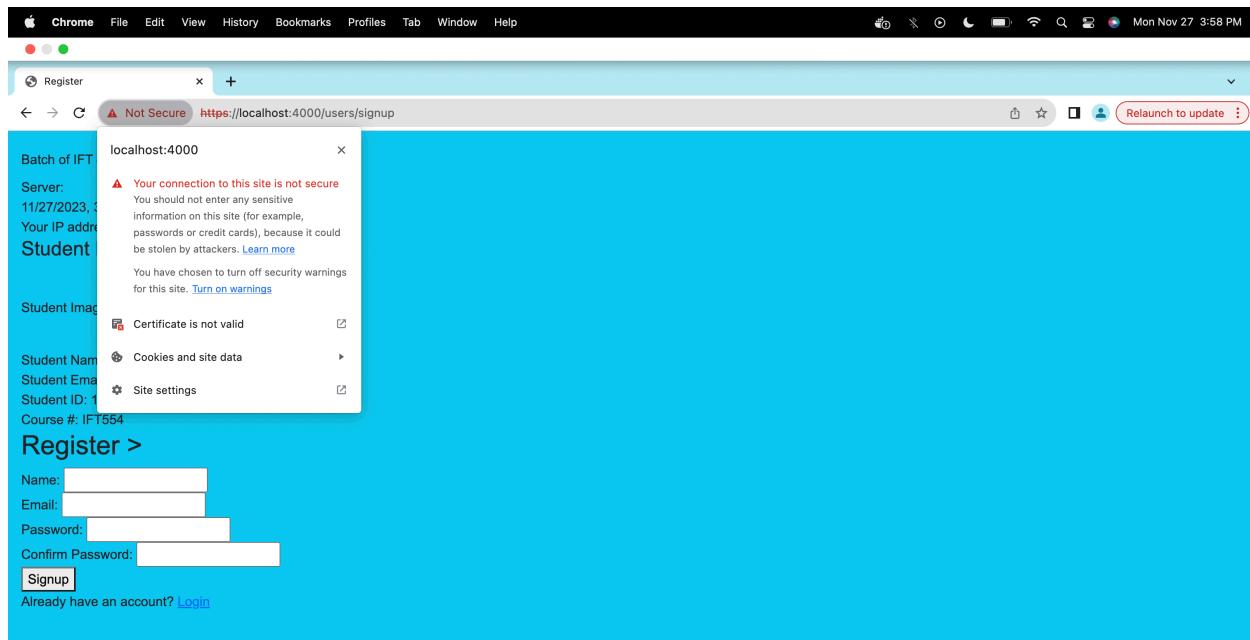
Accessing signup link: <https://localhost:4000/users/signup>



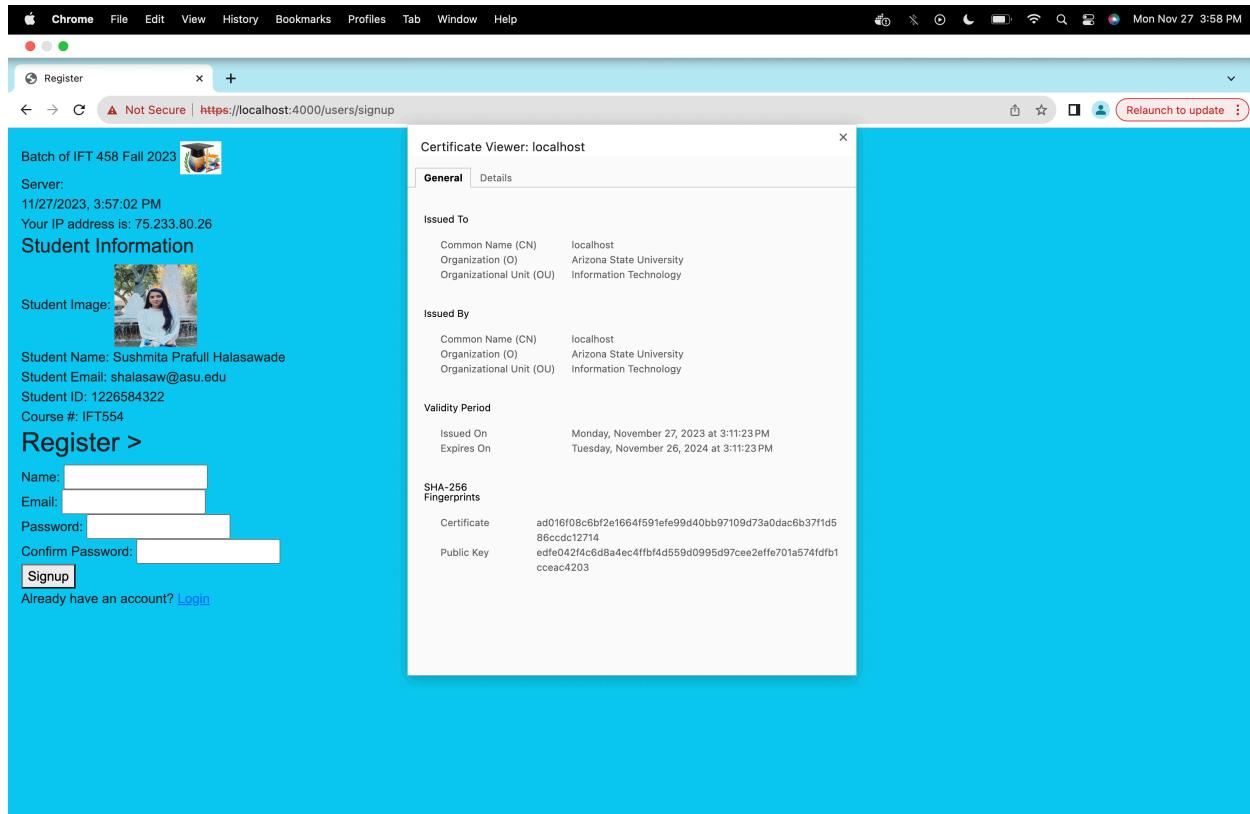
Proceed with localhost (unsafe):



Site is not secure because certificate is not valid:



Certificate and key details:



## Implementation:

### Challenges Faced During Implementation:

**Secure User Authentication:** Implementing a robust and secure user authentication system was crucial to protect user data and ensure privacy. The challenge lay in creating an authentication process that was both secure against threats and user-friendly.

**Efficient Data Management:** Transitioning from a basic array-based system to a more scalable MongoDB database posed challenges. The primary concern was efficiently managing a growing database of books and users, ensuring fast and reliable access to data.

Switching to HTTPS from HTTP: The transition to HTTPS was essential for enhancing security, particularly for protecting data during transmission. This required setting up SSL/TLS certificates and ensuring compatibility across all parts of the application.

#### Workarounds and Solutions Applied:

For Secure User Authentication: JWT (JSON Web Tokens) and bcryptjs were used for authentication and password hashing, respectively. These technologies provided a balance between security and performance, offering a reliable way to handle user sessions and protect sensitive data.

For Efficient Data Management: MongoDB, a NoSQL database, was integrated to handle the expanding data. It offered flexibility, scalability, and efficient handling of large volumes of unstructured data, making it a suitable choice for a dynamic platform like the Book Exchange website.

For HTTPS Implementation: The server was configured to handle HTTPS requests, ensuring all data transferred between the server and client is encrypted. This was vital for maintaining data integrity and confidentiality, especially for login credentials and personal user information.

#### Key Functionalities and Features:

User Registration and Authentication: Users can sign up and log in, with their credentials securely managed and authenticated.

Book Management: Users can add, update, delete, and browse books. This feature includes detailed information about each book, such as title, author, and exchange type.

Book Exchange Process: The platform facilitates the exchange of books between users, including the ability to request an exchange and view available books.

Security Features: Alongside secure authentication, the website includes middleware for route protection, ensuring only authenticated users can access certain functionalities.

## Learning Outcomes:

The development of the "Book Exchange" website provided an immersive learning experience, solidifying my understanding of several key concepts from Modules 5 to 8 and enhancing both my technical skills and broader comprehension of web development.

Technically, I gained in-depth knowledge of authentication and authorization mechanisms, particularly mastering the use of JSON Web Tokens (JWT) and OAuth. Implementing JWT for secure user sessions and OAuth for streamlined third-party authentications offered valuable hands-on experience, emphasizing the importance of robust security in web applications.

Transitioning the website to HTTPS using mTLS (Mutual Transport Layer Security) was another crucial learning curve, not only heightening my awareness of data security during transmission but also sharpening my skills in configuring and securing web servers.

The use of EJS for view designs introduced me to the dynamics of server-side rendering, blending JavaScript with HTML for dynamic content generation, significantly improving my design capabilities. Additionally, learning to test APIs with a REST client and utilizing RESTful APIs for efficient server-client communication were pivotal in understanding back-end to front-end interactions. The deployment phase using Docker offered insights into containerization and its benefits in ensuring consistency across different environments.

Navigating through these modules, I developed a keen sense of problem-solving and adaptability. Implementing JWT and OAuth, for instance, required not just coding skills, but an understanding of user behavior and security needs. Working with HTTPS and mTLS deepened my commitment to user privacy and data security. Designing views with EJS and testing APIs cultivated my attention to detail and the importance of user experience. Finally, deploying with Docker highlighted the significance of smooth and reliable application performance. This project was more than just coding; it was about creating a secure, user-friendly, and efficient platform that resonates with the needs of its users.

## Constructive Feedback:

- Modern Front-End Frameworks: The course mainly taught backend technology and how to render websites server-side with EJS. Including lessons on popular front-end frameworks like React, Angular, or Vue.js would give a more complete understanding of both front-end and back-end development, which is what full-stack developers are expected to know.

- Real-World Case Studies and Examples: Adding more examples and case studies from actual projects can help students understand how these technologies are used in real situations. This makes it easier to see how the theory they learn fits into practical work.
- Cloud Services and Scalability: Since cloud-based services are becoming more important, a section on cloud computing and how to manage big-scale applications would be really useful. This could involve learning about major cloud providers like AWS, Azure, or Google Cloud, and how to use Kubernetes for managing multiple application containers.
- In-Depth Backend Development: Expanding the parts of the course that deal with backend development would be useful. This should include more complex topics like how to manage large databases, building applications with many separate services (microservices), and making sure the server side of applications runs efficiently.

## References:

*JWT.IO - JSON Web Tokens Introduction.* (n.d.). Jwt.io. <https://jwt.io/introduction>

Mongoose. (2011). *Mongoose ODM v5.8.2*. Mongoosejs.com. <https://mongoosejs.com/ejs>. (2018, May 5). Npm. <https://www.npmjs.com/package/ejs>

*oAuth. (2020). OAuth 2.0 — OAuth.* Oauth.net. <https://oauth.net/2/>

Porter, B. (2021, February 7). *What is mTLS and How Does it Work?* Medium. <https://freedomben.medium.com/what-is-mtls-and-how-does-it-work-9dcdbf6c1e41>