Web Development Internship Program
Task 2: Beginner-Level E-Commerce Website

Objective:

The goal of this task is to build a simple e-commerce website focused on selling books. You will learn how to create a user login system, store user data, and perform basic product management. This project will help you understand the fundamental concepts of full-stack web development.

Instructions:

1. Develop an e-commerce website for selling books.
2. Implement a login and registration system to manage user data.
3. Display a list of books with options to view details and purchase.
4. Store user data and product information in a local JSON file or a simple database (like MongoDB or SQLite).
5. Submit the project as a ZIP file or upload it to a GitHub repository.
6. Include a detailed `README.md` file with setup and usage instructions.

Technologies to Use:

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Node.js, Express.js
- **Database:** MongoDB (or use a local JSON file for simplicity)
- **Package Manager:** npm
- **Authentication:** JSON Web Tokens (JWT) with `bcrypt` for password hashing

---

Project Requirements:

1. Project Title: Book Selling E-Commerce Website

---

2. Functional Requirements:

Frontend (HTML, CSS, JavaScript):

- User registration and login pages.

- Homepage displaying a list of available books with basic details (title, author, price).
- Product page with book details and a "Buy Now" button.
- Profile page to view and update user information.
- Responsive design using CSS frameworks like Bootstrap or Tailwind CSS.

Backend (Node.js, Express.js):

- Implement user authentication (register, login, logout).
- Manage book data (list, add, update, delete).
- Store user information securely with hashed passwords.
- Handle CRUD operations for books and user profiles.
- Use JWT for secure session management.

---

3. API Endpoints:

User Authentication:

- **POST** `/api/register` - Register a new user
- **POST** `/api/login` - Authenticate a user
- **GET** `/api/profile` - Get user profile data
- **PUT** `/api/profile` - Update user profile

Book Management:

- **GET** `/api/books` - Get the list of all books
- **POST** `/api/books` - Add a new book (admin only)
- **GET** `/api/books/:id` - Get details of a specific book
- **PUT** `/api/books/:id` - Update book details (admin only)
- **DELETE** `/api/books/:id` - Delete a book (admin only)

---

4. Project Structure:

```
graphql
CopyEdit
ecommerce-website/
├── server.js               # Backend server file
├── config/                 # Configuration files (e.g., database)
├── models/                 # Mongoose models (User, Book)
├── routes/                 # API route definitions
├── public/                 # Frontend files
│   ├── index.html          # Homepage
│   ├── login.html          # Login page
│   ├── register.html       # Registration page
│   ├── profile.html        # User profile page
```

```
│   ├── book.html           # Book details page
│   ├── css/                # Stylesheets
│   └── js/                 # Client-side JavaScript
├── data/                   # JSON files for storing data (if not using a
database)
├── package.json            # Project metadata and dependencies
└── README.md               # Project documentation
```

Example Features:

- **User Registration:** Sign up with a username and password.
- **Login/Logout:** Authenticate users and maintain session using JWT.
- **Book Catalog:** Display a list of books with prices and descriptions.
- **Profile Management:** Update user information like name and email.
- **Admin Features:** Add, update, and delete books (accessible only to admins).

Bonus Points:

- Implement a shopping cart to add multiple books for purchase.
- Display order history on the profile page.
- Add form validation and error handling for better user experience.
- Integrate payment gateway simulation (e.g., dummy PayPal or Stripe integration).
