Assignment Category: B12-A10_category-0001

Project Name: The Book Haven

Requirement Explanation Video

Project Theme

Build a full-stack web application where users can explore, add, update, and delete books in a digital library. Authenticated users will have their own control to manage book data. This project will demonstrate the student's ability to integrate a Node.js + Express.js backend, MongoDB database, and Firebase authentication into a single cohesive app.

Key Rules:

- **GitHub Commits:**
  - Include a minimum of 15 notable GitHub commits on the client side.
  - Include a minimum of 8 notable GitHub commits on the server side
- **Readme.md:** Add a meaningful readme.md file with the name of your website and a live site URL on client side. Include a minimum of five bullet points to feature your website.
- **Lorem Text:** Don't use any Lorem ipsum text; you can not use the default alert to show any error or success message.
- **Host your Application:** You can choose deployment systems like Netlify, Surge, and Firebase for client-side hosting and Vercel for server-side hosting. As you develop a single-page application
  - Ensure that the page doesn't throw any error on reloading from any routes.
  - Add your domain for authorization to Firebase if you use Netlify / surge
  - Logged in User must not be redirected to Login on reloading any private route
-

Main Requirements

Layout Structure

- Include a **Navbar** , **Footer**, and **Main Content Area**.

- The **Navbar** must have navigation links: Home, All Books, Add Book, My Books, and Login/Register.

Conditional Rendering:

- If the user is not logged in, show ”Login” and ”Register”.

- If the user is logged in, show their photoURL, displayName on hover, and a ”LogOut” button.

- The main section will be changed based on route.
- The **Footer** must contain copyright text.

Home

- Create a visually appealing homepage with:
  - **Banner:**
    Add a elegant banner with **animation, buttons (All Books, Create Book)**
  - **Dynamic Section:**
    Display the latest **6 books** added to the library, fetched dynamically from the MongoDB database.

  - **Two Static Sections (your choice):**
    Example ideas:
    - “Top Genres” – display book genres with images
    - “Book of the Week” – highlight a featured book with description and image
    - “About The Book Haven” – short intro about the site

Authentication

User Login:

- Create a Login page with a form containing:
  Email, Password, Forget Password (text only), and Login button.
- On successful login, navigate the user to the home page or their desired route.
- On failure, show a toast/error message.
- Add links for “Register” and a **Google Login button**.
- On successful Google login, navigate to the desired route/home page.

User Registration:

- Create a registration page with a form containing:
  Name, Email, Photo URL, Password, and Register button.
- Password validation rules:
  - Must include at least one uppercase letter
  - Must include at least one lowercase letter
  - Must be at least 6 characters long
- On successful registration, navigate to the home page.
- On failure, show a toast/error message.
- Include a link for “Login” and a **Google Login button**.

💡 **Do not** implement email verification or forget password for this milestone.

CRUD Operation

Routes:

- `/add-book` → Page to add a new book
- `/all-books` → Page to view all books from the database

- `/update-book/:id` → Page to update a selected book
-
  `/delete-book/:id` → Page to delete a selected book
- `/myBooks` → Page showing only books added by the logged-in user
- `/book-details` → Page showing only book details

Data Structure (stored in MongoDB):
Each book should contain:

```
{
  "title": "Book Title",
  "author": "Author Name",
  "genre": "Fantasy / Mystery / Non-Fiction etc.",
  "rating": "1–5 scale",
  "summary": "Short description of the book",
  "coverImage": "imgbb image URL",
  "userEmail": "email of the user who added the book"
}
```

Functional Requirements:

- Users can **Add** new books (with image upload via imgbb).
- Users can **Read** all books on the “All Books” page.
- Users can **Update** their own books (title, author, genre, rating, summary, cover image).
- Users can **Delete** their own books.
- Data must be stored and fetched from MongoDB Atlas.

All Books Page:

- Display all books in a table format.
- Show key info such as book name,author,genre, rating.
- Include a ”View Details” button to go to the book detail page.

View Details Page:

- Show full details of the book in a nicely designed format. Take inspiration from other book-related websites

- This will be a private route.

Add Book Page:

Create a form to add data to the database with the following fields:

- Title
- Author
- Genre
- Rating
- Summary
- coverImage

- User Email (who added this plant generally added user),

- User Name (who added this plant generally added user).

This will be a private route.
After successful submission, show a success message using any npm packages.

Note: Using browser alert is not allowed

My Books Page:

- Create a table of logged in user's added books.

- Show delete and update button in table. Make them functional.

Update Page/Modal:

- Create a pre-filled form with all editable fields.

- After submission, show a confirmation message upon successful update.

Other Requirements

- Show a **Loading Spinner** when data is fetching.
- Create a **Custom 404 Error Page** for invalid routes.
- Use **Axios** for data fetching.
- Protect routes like `/addBook`, `/myBooks`, and `/updateBook/:id`, `/deleteBook/:id` so only authenticated users can access them.

UI Design Requirements:

- **Unique Design:** First, decide what kind of website you want to make. Then, search online or check out websites like ThemeForest to get ideas for the design. But remember, your website idea shouldn't be similar to any projects you've done before or to any examples in our modules or conceptual sessions.

★ You can also look for free resources on blogs to help with your website.

1. Keep the main heading style (font, size, color) consistent across all sections.
2. Keep paragraph spacing balanced and text easily readable.
3. Maintain uniform image sizes and spacing.
4. Use the same button style as on the home page.
5. Ensure good spacing and proper alignment.
6. Navbar, Keep the heading/logo same style and size as on the home page.
7. Use a grid layout with equal image sizes.
8. Keep all cards equal height and width (especially in services, projects, or products section)
9. Use the new X logo instead of the old Twitter bird to match the latest rebrand
10. Responsiveness: Make it responsive for all devices, including mobile, tablet, and desktop views.

Resources:

- https://uiverse.io/

- https://devmeetsdevs.com/
- https://bootcamp.uxdesign.cc/free-images-and-resources-collection-for-website-c77f2fc46ce5
- https://themeforest.net/?srsltid=AfmBOopTj6PNz51iuV2YJXUtBP8nt19_zT5LG2dToAjIHQqzNCzregn0
- https://codecanyon.net/?srsltid=AfmBOooRoUfeK7IOROpchCuA4hPVj5P9WRmtDQJ9KOE6Yhf4VTrHhXKt

---

Challenges

1. **Advanced Filtering:**
   On the All Books page, implement sort functionality based on **Rating.** (Hint for sorting)

2. Add a **dark/light theme toggle** on the homepage.
3. Logged in user can add comments on book details page. Save the user name, photoURL and comment in database and show it in UI. Make sure it updates real-time.
4. Explore these packages and implement at least two:

   a. React Hot Toast

   b. Date fns

   c. React Tooltip

Optional Requirements

1. **Review Section:**
   Let users leave short text reviews on each book page. Reviews should be stored in MongoDB and displayed dynamically.

2. **Top Rated Books Section:**
   On the home page, display the top 3 books with the highest ratings (use MongoDB's `$sort` operator).

What to Submit

- **Client-side GitHub repository link**
- **Server-side GitHub repository link**
- **Live website link**