WALKERS

Walkers Technical Assessment – Full Stack Developer Assessment

Sharing Your Solution

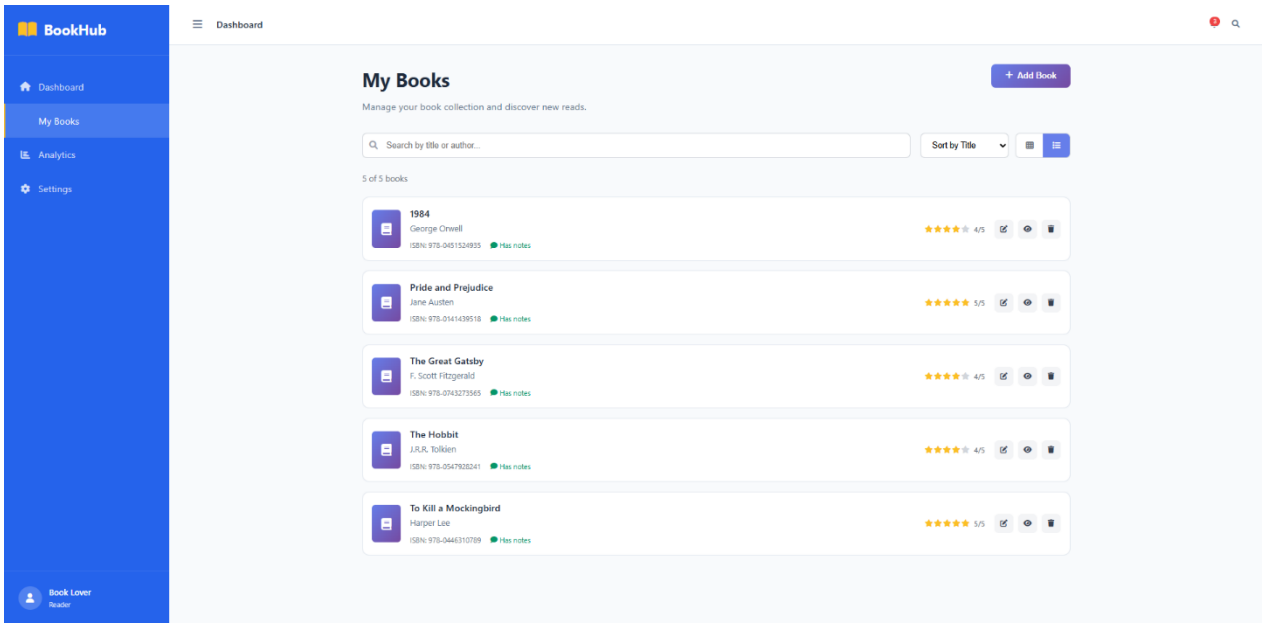
Please use github.com and share your solution with the account **jaydobson**. Should you have any issues or questions with this assignment please reach-out to jay.dobson@walkersglobal.com

Scenario

Given the visual reference for a simple book-tracking application below, your task is to recreate this UI in Vue.js, using modular, reusable components and clean, production-quality code.

Design Reference

The following image represents the style, layout, and key features you should implement.



Requirements

1. UI Implementation

- Use Vue 3.
- Reproduce the main “My Books” screen, including:
  - Sidebar with navigation.
  - Main content card with:
    - Page title
    - Description
    - “Add Book” button (opens a modal form)
    - Search bar (“Search by title or author…”)
    - Sort by Title dropdown

- Option to switch between grid/list view (see icons, but only list view is required)
- List of books: each showing a cover/thumb, title, author, ISBN, note status, star rating, and action buttons (edit, view, delete)
- Pagination at the bottom if needed

Forms & Modals

- Add Book modal: collect title, author, rating (1–5), comments, ISBN.
- Edit Book modal: can only edit rating and comments (other fields are read-only).
- Delete confirmation modal: “Are you sure you want to delete a book/review?”

Additional Freestyle Pages

- The sidebar in the reference UI includes links to Analytics and Settings pages.
- For these pages, you are encouraged to use your creativity:
  - Analytics: Show any stats, charts, or data visualizations related to the books collection that you find interesting. You may use mock/generated data.
  - Settings: Provide a basic settings/preferences page—feel free to decide what options to include (e.g., profile info, theme toggle, etc.).
- The content, layout, and design of these pages are completely up to you.
- Bonus: Briefly explain your choices in your README or VISUAL_IMPROVEMENTS.md.

2. Backend API Implementation (Required)

- Implement a REST API in .NET (C#) with in-memory storage (no database required, but a nice to have) to support all the necessary operations for the Book List application:
  - Add a book
  - Edit a book (only rating and comments are editable)
  - Delete a book
  - Get all books (with pagination, search by title/author, sort by title)
  - Get a single book (by ID)
- Store all required fields: Title, Author, ISBN, Rating, Comments, Note status, and Cover image URL(s) in memory (e.g., a list or collection).
- Enforce all business logic and validation rules on the backend as described for the frontend (e.g., max 25 books, rating 1–5, “horrible” disallowed in comments, sensible string length limits).
- Implement proper error handling and return meaningful HTTP status codes/messages.
- Include at least one unit test for the API (using xUnit, NUnit, or MSTest).
- Provide setup instructions for running the backend locally (README).

3. Functional Requirements

- Implement search (by title or author), sort (by title), and pagination (show 10 books per page if many).
- Implement Add, Edit, Delete, and View Details functionality.
- Display a notes indicator if a book has comments/notes.
- View Details screen: show all book info, cover, rating (with stars), comments.

4. Validation

- Max 25 books in total.
- Rating: must be 1–5.
- Comments: required if rating is supplied; must not contain the word "horrible".

- String fields: enforce sensible character limits (in form and code).
- All validations must show user-friendly feedback.

5. Code Quality

- Modular Vue components, clear naming, no commented code, consistent formatting.
- README with setup instructions.
- At least one unit test (e.g., with Vitest or Jest + Vue Test Utils).

6. UX & Visual Polish

- Responsive layout (works on mobile/tablet).
- Use hover/focus effects for action buttons.
- Professional design matching the reference (color, spacing, icons, font).

7. Documentation

- Include a short VISUAL_IMPROVEMENTS.md describing any UX/UI improvements you would suggest for a real product.

Deliverables

- GitHub repo with all code.
- README.md with setup instructions.
- Live link if possible
- VISUAL_IMPROVEMENTS.md with your feedback.
- At least one unit test.

Bonus (Optional)

- Implement grid view.
- Use a UI framework (e.g., Vuetify, Element Plus).
- Animations/transitions for modals and action feedback.
- Advanced validation or accessibility enhancements.