Book Tracker

Personal reading list manager

Sanjana Chowdary

Problem

- I found it challenging to recall which books I should start, continue, or finish.
- I often overlooked books that I wanted to read or had been recommended to me.

What I wanted

- Wanted lightweight stats without a spreadsheet
- Quick way to record thoughts/ratings of books

Solution

Reading List Tracker

Search and **Store** (shelves, sorting, ratings, stats)

search

store

Book List API

Pulls book data from Open Library so you can add books fast

When a Book is searched and added, you also have the ability to rate an already read book along with other status

Persisted locally so your list is always there

Have 3 main lists

1. Want to read
2. Reading
3. Read

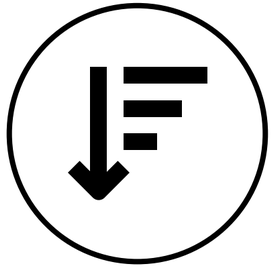
Display statistics about your reading habits and also view your ratings of read books

Core Features

Add books via Open Library search

Shelves: Want to Read / Currently Reading / Read

1–5 heart ratings for finished books



Search + sort (date, title, author)

Reading stats widget (totals, average rating)

Architecture

- *React + Redux Toolkit for book state (add/update/delete/status/rating)*
- *Context API for reading stats (shared without prop drilling)*
- *Custom hook useBooks combines Redux and localStorage persistence*
- *Axios -> Open Library API for book search*

LIVE DEMO!

Final > book-tracker > src > hooks > JS useBooks.js > ...

```
14  import { loadBooksFromStorage, saveBooksToStorage } from '../services/localStorage'
15
16  export const useBooks = () => {
17    // Using useSelector and useDispatch from react-redux - learned these in class
18    const dispatch = useDispatch()
19    const books = useSelector(state => state.book.books)
20
21    // Load books from localStorage when component mounts
22    // This useEffect pattern I learned from class - runs once on mount
23    useEffect(() => {
24      const savedBooks = loadBooksFromStorage()
25      if (savedBooks.length > 0) {
26        dispatch(setBooks(savedBooks))
27          }
28    }, [dispatch])
29
30    // Save books to localStorage whenever the books array changes
31    // This was interesting - automatically persisting state changes!
32    useEffect(() => {
33      if (books.length >= 0) {
34        saveBooksToStorage(books)
35      }
36    }, [books])
37
38    // Function to add a new book with default values
39    const handleAddBook = (bookData) => {
40      const newBook = {
41          ...bookData,
42        id: Date.now().toString(), // Generate unique ID
43          dateAdded: new Date().toISOString(), // Track when book was added
44        status: bookData.status || 'want-to-read', // Default status
45        rating: bookData.rating || null,
46        notes: bookData.notes || '',
47        }
48          dispatch(addBook(newBook))
49          return newBook
50    }
```

What was Hard

- Combining Redux + localStorage: two useEffects to load on mount and save on every change without double-dispatches.
- Building new book objects with defaults: generating ids, dateAdded, default status/rating/notes before dispatching.
- Keeping state in sync: making sure add/update/delete/status/rating all flow through Redux and persist correctly.

UI & UX Notes

- Sidebar navigation with “Continue Reading.”
- Shelves are horizontally scrollable; “All Books” grid with sort dropdown
- Soft color palette (lavender/mint/warm-brown), matching search/sort controls
- Animated quote in stats for a bit of delight

What was Fun

01

useSelector to grab current reading book for the sidebar “Continue Reading”

02

Building the animated quote fade-in/out

03

Seeing stats auto-update when shelves change

Code I’m Proud Of

useBooks hook

cleanly wraps Redux + localStorage; one place for add/update/delete/status /rating

useSelector

in App.js for “Continue Reading” sidebar which is simple, direct state access

ReadingStats + Context

memoized stats and animated quote without prop drilling

Sorting Logic

In “all books”, this handles date/title/author with sensible fallbacks

How I Grew as a React Dev

- Learned when to use Redux vs Context (state vs derived data)
- Got comfortable with custom hooks to share logic across components
- Improved data mapping from external APIs (Open Library) into app-friendly shapes
- Made the UI flow smoother: search, add, move, rate, see stats; kept styling consistent, shelves scrollable, and added small touches like the animated quotes.

Thanks!