Zdenek Andrew Bocek

January 4, 2024

# My Project Proposal: Look Up Book

1. I will use NodeJS, Express, React, React Router, Babel, HTML, CSS, and possibly some other tech stacks to complete my project.
2. The front-end UI will be the focus of my project since I will be using react and react router. All requests should be made through the front end in the browser, instead of retrieving requests from the server side.
3. This will be an app that should auto fit any screen it is rendered on.
4. I will make an API and call it "Look Up Book". This will not be a Library API, but rather, it will be one that holds details on authors of books, how many literary prizes they have won, details of the books they have written, what year the books came out, etc. It will be a recommendation engine for people looking up genres of books, or regions of the world where books have been written, or authors of books, or what sort of award they want the book they read to have won. It will not feature authors who have simply been put on "bestseller lists", because all an author needs to do to end up on a bestseller list is have most American consumers in a certain city read their book. The New York Times bestseller list simply tells you that an author's book was read by most people who live in New York City, but IT DOES NOT tell you whether that author won any literary prizes at all for their book. You don't need to win any awards whatsoever to end up on a bestseller list as an author. My API will serve as a recommendation engine for people who are looking to read books by award-winning authors. These awards will be serious awards that require real achievements in writing to win, including the Nobel Prize in Literature, the Pulitzer Prize, the Booker Prize, the International Booker Prize, PEN America Literary Awards, the National Book Award, the Costa Book Award, or the PEN Faulkner Award. They can search for books using multiple metrics, and they can post the books they choose to their own account or put them on a shopping list. They can also update their lists and delete books from their lists.
5. I predict that the demographic that will use my app will be from almost every age group, every ethnicity, every race, and probably every gender. It is up to readers if they want to read a book that has been rated by other literary professionals as being above and beyond in terms of creativeness and intelligence over a book that has simply appeared on a bestseller list.
6. I will create my own API and populate it with data. I will try to collect data from libraries, as well as any other place where I do not infringe on copyright.
7. a. My database schema will have the user table at the center, from which the user will be able to send a get request to multiple tables from my API. My database will have several many-to-many as well as one-to-many relationships.

b. I will probably want to avoid a race condition between data in different tables. I also will probably want not more than one piece of data in one bucket. If I must store multiple data pieces in one bucket, I will probably want to make sure that my app doesn't get confused about which piece of data to send.

c. I will secure all users’ passwords, as well as the secret key to connect to my API.

d. My app will use POST, GET, DELETE, and PUT. In short, it should use every aspect of the CRUD cycle.

e. The user will land on the home page, where he/she will need to either log in or create a new account. After that, he/she will be taken to his/her own profile page, where it should show what books he/she has already stored in his/her account. He/she will be able to request a new search for a book he/she wishes to read. The user will be able to select a new book based on the author, a country of origin, professional awards that the author has won, genre of book, year that the book was published, etc.

f. I think this goes beyond being a simple CRUD app in terms of its creativity and originality. I don’t think anyone has tried to offer readers an app where they can look for books from authors that have done more than just being popular.