Authentication

1. The main challenge was implementing the ownership tracker. I needed to modify the database scheme to add “create_by” columns to both “authors” and “books” tables. I did this by updating the existing setup.sql and making sure all “Insert” statements included the user ID.
2. The hard part was deciding when to show versus hide UI elements. At first, I made the entire app require a login to see everything, but then I decided that the views should be able to see the books that are in the library. And if they wanted to add/delete anything, they would have to log in.
   Note: I did use an LLM to help me with the design aspect of this assignment and to help me generate authors and booksto put into my library.

Deployment:

1. I struggled with understanding how to serve the compiled React frontend from node backend rather than using the Vite development server. Moving the built frontend into the backend “dist” took lots of trial and error.

Security audit:

1. My app was not vulnerable to reflection XSS attacks because the user data is never injected into the DOM as raw HTML. Because React automatically escapes values rendered in JSK, which stops bad scripts from being run.
2. My app was vulnerable to CSRF attacks because I used default cookie settings without CSRF protection. This meant if a user logged into my app and visited a malicious website, the website could make requests to my API endpoint without the user's cookies, attacking the browser. I fixed this by adding “sameSite: ‘lax’ ” to all authentication cookies. This makes sure browsers only send cookies with requests that originate from the same site.
3. I added rate limiting in my application using the express-rate-limit package. Login routes are protected with stricter limits, which help prevent brute-force attacks.
4. I used the Helmet middleware to set security headers. Content-Security-Policy to monitor where scripts and styles can be loaded. X-Frame-Options to prevent clickjacking. And x-Content-Type-Options to stop MIME-type sniffing.
5. Passwords are hashed using Argon2 before they are stored in the database.
   Note: I did use an LLM to help me understand what all these security issues were and how to deal with them.