CS 348 Project Report – Shreya Venkatraman

Github link - https://github.com/Shreyav1231/Libgenie

Libgenie is a dynamic and robust app intended to be used by a librarian to manage their small bookstore.

Technologies used –

Platform & language – Node.js, JavaScript

Web framework – Express.js

Database access – MySQL, Sequelize ORM

Frontend – React, Axios

To start the app -

cd server

npm start

cd client/my-app

npm run dev

1. How to use –

On starting the app the user will see all of the loans. User has the option to view authors, view books or generate report.

A screenshot of a computer

Description automatically generated

User can either delete an existing author or add a new author –

A screenshot of a phone

Description automatically generated

User can add new books – the dropdown of authors gets dynamically built from the authors table so it shows only existing authors

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

User may filter books within a date range and can view the books that have been taken out within that date range and haven’t been returned yet. The app correctly shows the number of days the book has been out.   
Bestseller author is an author who has written the most number books that haven’t been returned yet.

A black and white screen with white text

Description automatically generated

1. Indexes report –

const indexStmts = [

`CREATE INDEX ix\_loans\_return\_borrow\_date ON loans (return\_date, borrow\_date)`,

`CREATE INDEX ix\_loans\_book\_id ON loans (book\_id)`,

`CREATE INDEX ix\_loans\_borrower\_id ON loans (borrower\_id)`,

`CREATE INDEX ix\_books\_author\_id ON books (author\_id)`

];

CREATE INDEX ix\_loans\_return\_borrow\_date ON loans (return\_date, borrow\_date);

By having return\_date first, the optimizer can quickly locate rows where return\_date IS NULL

This index is very useful for the report based on finding books borrowed within a range of dates.

CREATE INDEX ix\_loans\_book\_id ON loans (book\_id);

This index speeds up lookups on loans where you filter or join by book\_id – useful in returning the book name for the report

CREATE INDEX ix\_loans\_borrower\_id ON loans (borrower\_id);

Without this index, fetching all loans for a particular borrower would force a full scan of loans.

CREATE INDEX ix\_books\_author\_id ON books (author\_id);

This index helps speed up the join from books to author when grouping or filtering on author\_id.

As this app is only intended for a single librarian’s use, I have not added concurrency. However, in the future here’s how I will implement it –

1. Simultaneous check-out requests – two clerks could try to loan out the same book at once. I would wrap in a transaction so only one INSERT + UPDATE pair succeeds.
2. Bulk admin operations – admin scripts that delete or archive old loans shouldn’t leave the system half-updated if they fail midway.
3. Running a best-seller calculation at the same time as users checking out their books could lead to phantom reads. We must then choose a higher isolation level to guarantee consistency.

Lessons learnt –

Building both a frontend (React) and a backend (Express/Node.js) taught me to clearly delegate responsibilities - The frontend focuses on user interaction, rendering UI components, and invoking API calls. The backend handles the data management, indexing and security. I learned how data flows from the database, through Express endpoints, as JSON and finally into React components, and then back again when the user fills the fields in and submits.