Library Route Finder

Release 1.0

Isaac List

May 03, 2022

CONTENTS

The **Library Route Finder** is a web-based tool to assist in finding items in Preus Library at Luther College. The application is composed of a React-based frontend, as well as an API-based backend.

Check out the *For Developers* section for technical information about the project, and the *API* section for information about the API provided by the backend. For contributing and community guidelines, please see *Contributing*.

This project was developed as a Senior Project for the Luther College Computer Science department.

The stated purpose of this course is to familiarize students with the process of developing a software project from conception through development and testing and to deployment of a final product. In that spirit, our group aspires to use our technical and organizational skills to create a web-based application which assists users in locating books within a library, providing a visual map and route showing both item locations and an efficient route to those items.

Contributors:

- Firdavs Atabaev
- Alex Dikelsky
- Isaac List

CHAPTER

ONE

FOR DEVELOPERS

1.1 Installation: Development Environment

The development environment and setup for each component is described in reasonable detail in the relevant repository's README file. The documentation is reproduced below:

1.1.1 General: Cloning and Setting Up the Node.js Environment

Both the Frontend and Backend utilize Node.js as a platform, the latter also using React as its primary framework. The process for cloning the repository and installing Node dependencies is the same between both components:

Cloning the Project:

To contribute to the project, it is expected that you first create a fork of the relevant repository, clone that repository, perform your work, and use the Pull Request mechanism to contribute to the main repository. Github has easy-to-follow documentation on this process available here:.

Installing Node dependencies

To run the project component(s) locally on your machine, you must first install the NPM packages upon which the component(s) depend. This must be done for each component which has its own repository. First, ensure that Node and NPM are installed (it is recommended using NVM, the Node Version Manager). Then, run npm install in the root of the repository's directory. This will install the packages recorded in package.json as dependencies.

To run the project locally, enter `npm start`.

1.2 Components

1.2.1 Frontend

The Frontend utilizes React as its main framework, and as such must be built before deployment. This build may be performed by running npm build in the project's root directory. This will produce a `/build` directory from which Netlify deploys. This directory contains the transpiled JavaScript, HTML, and CSS which is created from the main project code.

1.2.2 Backend

The Backend uses Express.js as its main framework, and unlike the Frontend, does not need to be "compiled" in the same sense before deployment. However, the main "App" component is written in TypeScript, which does need to be compiled before deployment. This process is completed automatically when the npm start script is evoked,a process which is also completed by Heroku when the project is deployed.

1.3 Testing

1.3.1 Frontend

Once testing is implemented later in the semester, a script will be defined which can then be invoked with `npm test`.

1.3.2 Backend

The testing script for the backend is defined in the package.json file as being invoked by npm test. This runs the test file defined at `/test/test.js`. This file uses the Mocha testing library to test the backend's routes and return values. This script is run each time a pull request is merged into the main repository.

The Backend's API is tested using a set of Postman tests.

CHAPTER

TWO

API

2.1 Routes

The API consists of 3 main routes defined in the backend's `app.ts` file:

2.1.1 Books

Located at `/api/books`, this route accepts `GET` requests with the query parameter of "`?name=<username>`" where `<username>` is any username present in the application database.

This route returns the list of books associated with a given username as JSON in the following format:

```
{
  "results": [
    {
      "isbn": "9781611321456",
      "author": "Author of the Work",
      "title": "Title of the Work",
      "call_no": "AM101.L196",
      "username": "username"
    },
    {
      "isbn": "9781611321456",
      "author": "Author of the Work",
      "title": "Title of the Work",
      "call_no": "AM101.L196",
      "username": "username"
    }
}
```

2.1.2 Search

Located at `/api/search`, this route accepts `POST` requests with the following header parameters:

- isbn: a valid ISBN-10 or ISBN-13 code
- name: a username present in the application database

Passing a request to this endpoint will result in the requested item being added to the application database. The following JSON will be returned in the case of either:

Success

`{"status": "success", "book": item}` where `item` is the book information.

Failure – Database Issue

`{"status": "failure", "error": data}` where data is the response returned from the OCLC access module.

Failure – Bad Input

`{"status": "failure", "error": "Invalid ISBN"}`

2.1.3 Remove

Located at `/api/remove`, this route accepts `POST` requests with the following header parameter:

- name: a username present in the application database

Passing a request to this endpoint will result in all items associated with that username being removed from the database, with the end-user result of clearing the subject user's list of books. It will return `{"Status": "Success"}` if the removal was successful, or `{"Status": "Failure", "Error": err}` in the case of an internal error, where `err` is any error reported by the database.

2.2 Database Interaction

Use of the API requires that the PostgreSQL database connection be active. The Backend will fail to launch if the database connection is configured incorrectly. Refer to *Database* for more information.

CHAPTER

THREE

DATABASE

3.1 Technologies

3.1.1 PosgreSQL

This application uses the PosgreSQL database in its implementation. As currently deployed, the project uses the database option provided by Heroku's platform.

3.1.2 Node-Postgres

The backend is built using the Node-Postgres object relational mapping module to interact with the provided database.

3.2 Database Structure

The database uses a single table in the following configuration to store all books or other materials added to users' lists:

Columns:

- isbn: 10 or 13-digit string
- author: up to 64 character string of author name
- title: up to 256 character string of item title
- call_no: up to 48 character string of item call_no
- username: up to 64 character string of item's associated user

Primary Key:

The table's primary key is the combination of the values isbn and username.

3.2.1 SQL Configuration

The following SQL command will create the necessary table for the application.

```
create table booklist IF NOT EXISTS (
  isbn VARCHAR(16),
  author VARCHAR(64),
  title VARCHAR(256),
  call_no VARCHAR(48),
  username VARCHAR(64),
  PRIMARY KEY (isbn, username)
);
```

CHAPTER

FOUR

MAPPING

4.1 Application Purpose

The purpose of the mapping program is to provide information for reshelving books. It does this by providing a map of the library, in addition to highlighted shelves that have books in need of being reshelved.

Developers can extend the given map for their own purposes if needed by modifying *map.json* with new data as described below.

4.2 Creating a new map

Map.tsx, in the src/components/ui directory reads from a file in the same directory called *map.json* for the creation of the map. The layout of the json in *map.json* is

```
Array<
   {
   "left": Array<{"min: string, "max": string}>
   "right": Array<{"min": string, "max": string}>
}>
```

The outer Array is the entire list of rows in the library. The *"left"* and *"right"* object are for the left and right sides respectively of the row. The next-inner-most array is the list of shelves contained in a row, which each have minimum and maximum library codes.

The strings should be library codes should be in a form such as *QA1*, in a way that is readable for our ordering scheme.

To use this new map, overwrite the contents of *map.json* with the above JSON.

4.3 Modifying minor aspects

To change the colors of the books, modify the constants *NO_BOOKS_COLOR* and *HAS_BOOKS_COLOR*. You can also modify the width between the shelves, and certain other aspects of the map using the global constants at the top of the file.

4.4 Modifying the code

If the map produced looks incorrect in some way, such as being too long for the screen, you may need to use or write your own functions. The idea of the code at the time of writing is to be extremely modular and mutation-free so that the code will be not as complicated as it could be to understand.

The most important thing to understand about the code is the type structure. All of the interfaces are at the top of the file, and explain in detail what everything looks like.

4.5 React components

- *Shelf*: Shelves are building block of rows, and need information, such as the color of each of their sides, their x,y coordinates, and the range of books they contain.
- *Row*: Rows contain lists of shelves that will be rendered with the first element shelf at the top.

CHAPTER

FIVE

CONTRIBUTING

5.1 Filing Issues

5.1.1 Create a new issue

If you spot a problem with the component, search if an issue already exists. If the issue you've encountered has not yet been documented, you can create a new issue using the appropriate issue template on GitHub.

5.2 Contributing Code

5.2.1 Getting Set Up

Cloning the Project: To contribute to the project, it is expected that you first create a fork of the repository and clone that repository to your machine. Github has easy-to-follow documentation on this process.

Installing Node dependencies To run the project locally on your machine, you must first install the NPM packages upon which the project depends. First, ensure that Node and NPM are installed (we recommend using NVM, the Node Version Manager). Then, run `npm install` in the root of the repository's directory. This will install the packages recorded in package.json as dependencies.

To run either component the project locally, enter `npm start`.

5.2.2 Pull Request

When you're finished with the changes, create a pull request, also known as a PR.

1. Push all local commits to your fork of the repository.
2. On your fork's main GitHub page, click "Contribute" and then "Open Pull Request".
3. Give your PR a title, and briefly describe the changes you have made. Keep each pull request limited in its scope, so that changes are more modular.
4. If necessary, or if you would like help with your work, request a Reviewer (see Code Review below).

5.3 Code Review

5.3.1 Code Review Procedure

Consider whether you should request a review of your code from another contributor. You should request a manual code review if:

- One or more unit tests are failing
- The PR addresses a long-standing bug or introduces new behavior
- The code uses constructs unfamiliar to the other contributors

5.3.2 Code Review Etiquette

CSS Tricks has a well-written guide to maintaining respectful etiquette in a Code Review process. The article is worth a read-through, but in summary:

1. Remove the person: use "we" instead of "I" and "you" to reflect that reviewing code is a collaborative activity.
2. Keep conversation focused on technical problems and solutions. Avoid emotional responses, and instead use clarifying questions to direct discussion.
3. Review the code, not the author.
    - Programming is as creative as it is technical, and each person approaches it differently.
    - Where possible, seek to correct mistakes by teaching, rather than dismissal.
    - If there is a conflict about coding style, refer to the project Style Guide.

CHAPTER

SIX

FOR USERS

6.1 Application Purpose

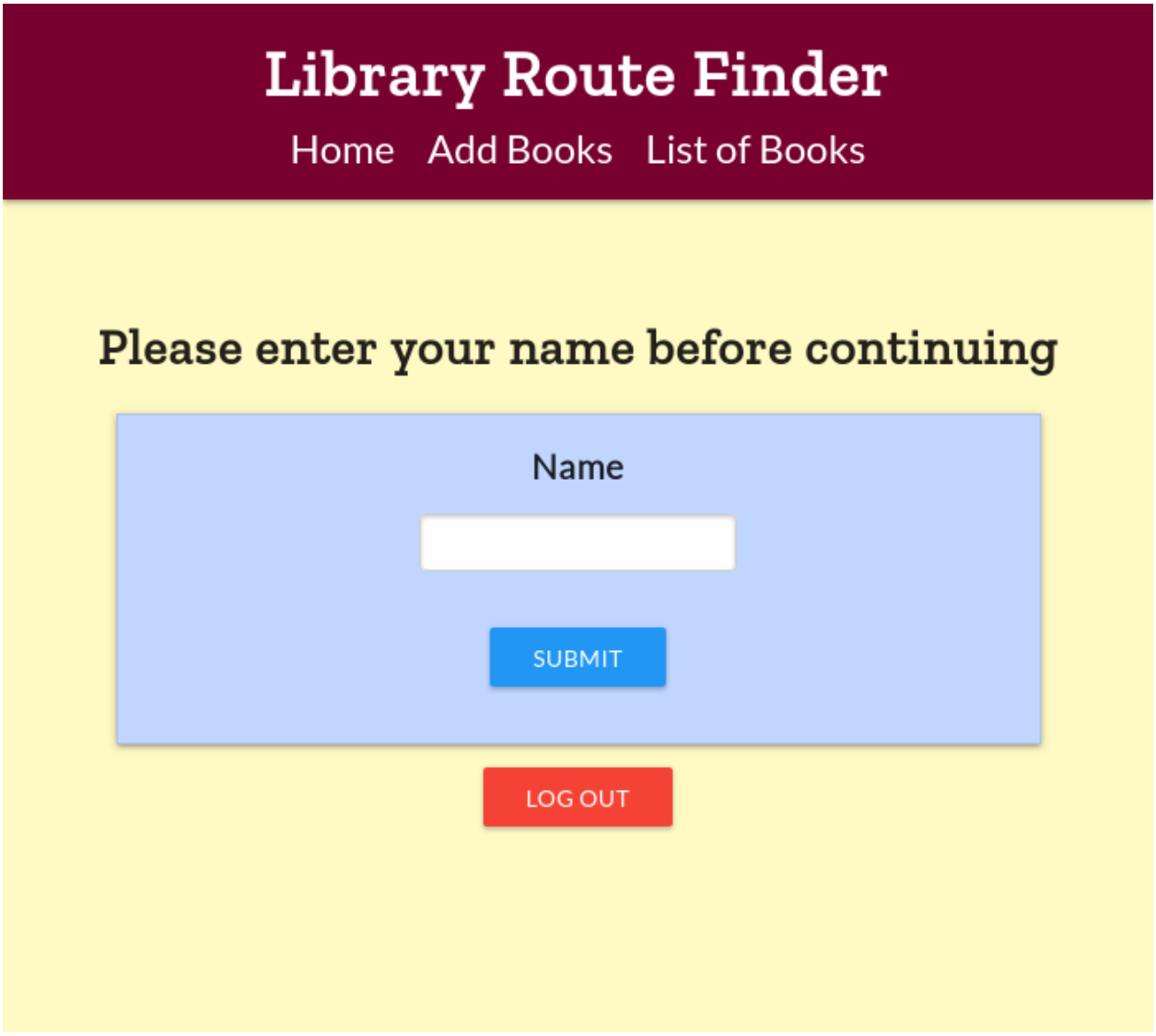
The purpose of the **Library Route Finder** is to make the retrieval of items in the library. The deployment our team maintains (available here) uses the floor plans of Preus Library at Luther College for the purpose of demonstration. Our stated goals for this project include improving the experience of locating library materials, as well as minimizing distance to walk if a patron or employee is visiting many shelves.

In addition to seeking to improve these real-world use cases in the realm of library patron experience, we also envisioned this project as implementing a portable concept. The idea of making the location of items within a library, store, warehouse, etc. has positive potential for a wide range of people in the areas of accessibility, orientation, and efficiency.

6.2 Getting Started

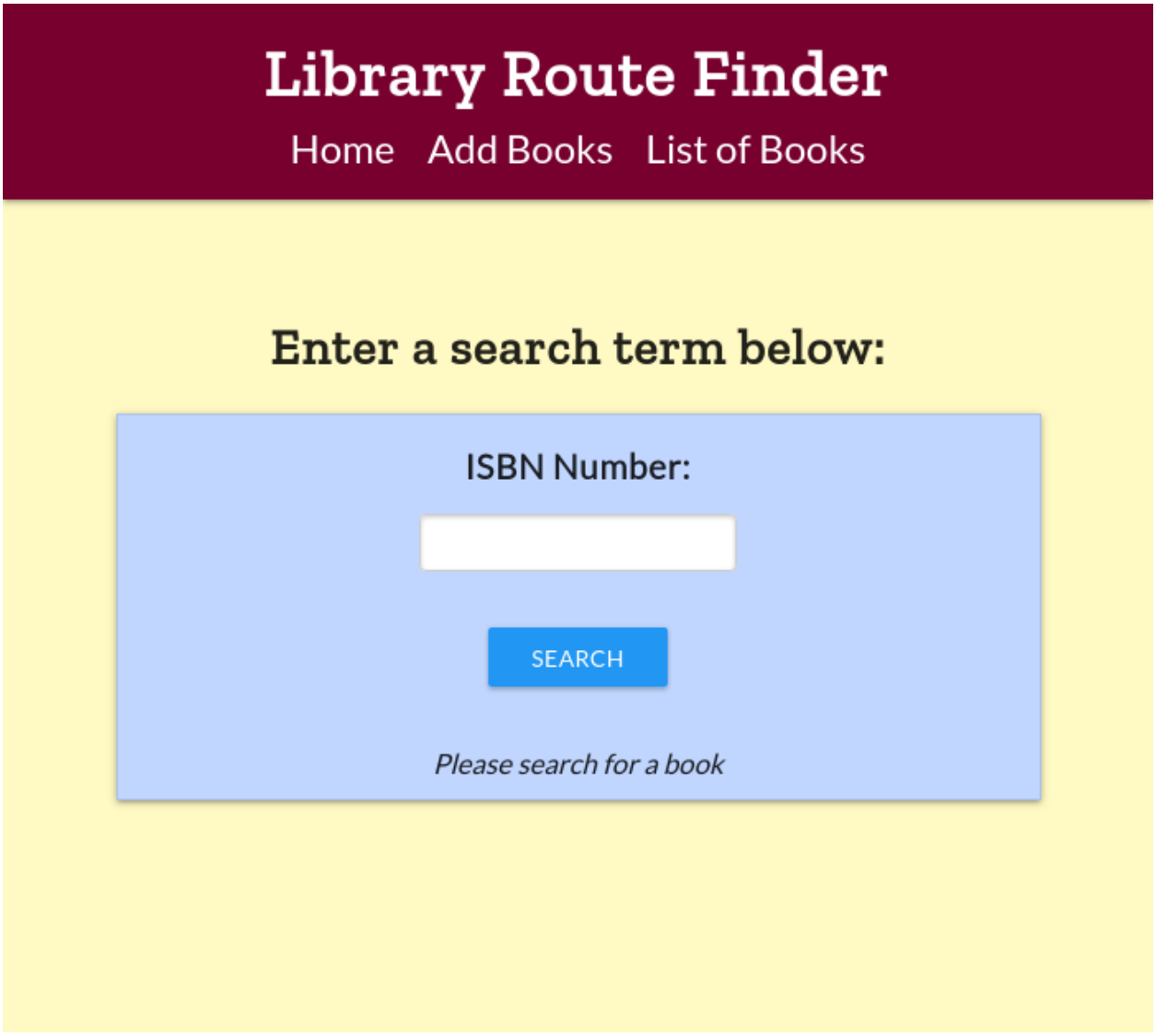
6.2.1 Logging In

The first step in using this program is to enter your name (or a username of your choice) into the form on the home page:



6.2.2 Searching for a Book to Add

When you click "Log In" on the first page, you will be redirected to the Search page. On this page is a form similar to the Login form, this time requesting an ISBN (International Standard Book Number).



To add a book, simply enter its ISBN in the box provided, and click "Search". The text below this button will update with a message indicating what book was added to your list, or in the case of an error will communicate the issue.

6.2.3 Viewing the Booklist and Route

To view your list of books as well as a map of which shelves you will have to visit to retrieve them, visit the "List of Books" link in the navigation bar. This will bring you to this page, where you will see your list outlined:

Highlighted in dark blue are the shelves you should visit, while the light blue shelves indicate ones to skip. Multiple books may be located on the same shelf, especially for books on the same topic, so the number of highlighted shelves is likely to differ from the number of items on your list.

Clearing the List

To clear your list of items, simply click or tap the red “Clear List” button. This action cannot be undone, and you will have to return to the “Add Books” page in order to re-locate any items.