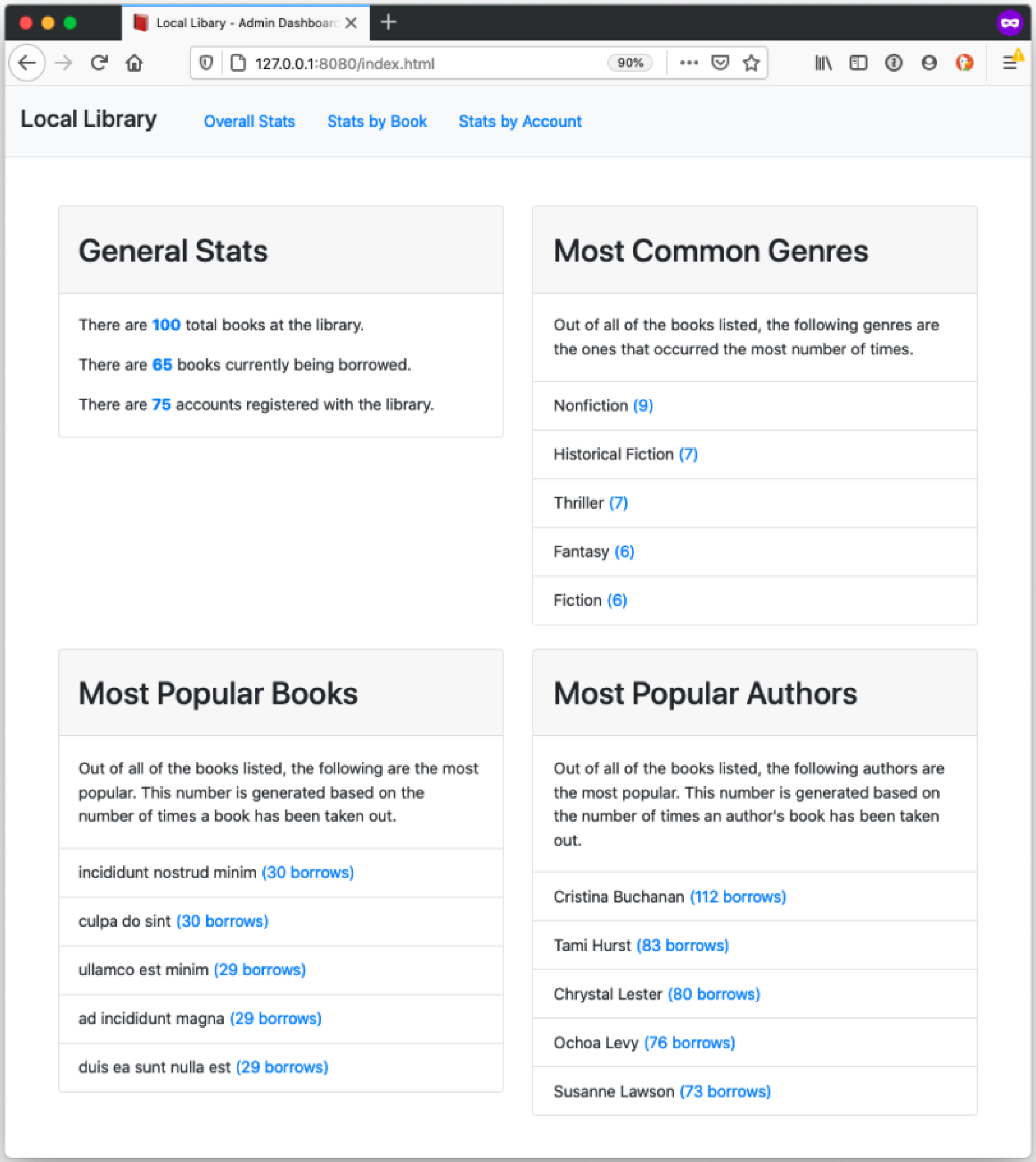
Your neighborhood has decided to put together a local library where people can lend and borrow books. One of the most important features to organize this is a dashboard that will show which books are available, which are currently out, and other general statistics about the program.

Others are taking care of the logistics and and design, but they need you to build the algorithms!



General Stats

There are 100 total books at the library.

There are 65 books currently being borrowed.

There are 75 accounts registered with the library.

Most Common Genres

Out of all of the books listed, the following genres are the ones that occurred the most number of times.

Nonfiction (9)

Historical Fiction (7)

Thriller (7)

Fantasy (6)

Fiction (6)

Most Popular Books

Out of all of the books listed, the following are the most popular. This number is generated based on the number of times a book has been taken out.

incididunt nostrud minim (30 borrows)

culpa do sint (30 borrows)

ullamco est minim (29 borrows)

ad incididunt magna (29 borrows)

duis ea sunt nulla est (29 borrows)

Most Popular Authors

Out of all of the books listed, the following authors are the most popular. This number is generated based on the number of times an author's book has been taken out.

Cristina Buchanan (112 borrows)

Tami Hurst (83 borrows)

Chrystal Lester (80 borrows)

Ochoa Levy (76 borrows)

Susanne Lawson (73 borrows)

This project is designed to test your ability to work with large datasets and build algorithms based on those datasets. Before taking on this module, you should be comfortable with the learning objectives listed below. You will not need to make any edits to HTML or CSS for this project.

Learning objectives

This project will assess the following learning objectives, in addition to many others:

- Use Visual Studio Code as a text editor.
- Differentiate between the three most common JavaScript error types.
- Solve bugs by using error messages.
- Differentiate between `let`, `const`, and `var`, and use each appropriately.
- Create an array from a string based on particular criteria and join arrays into strings.
- Write strings that embed expressions using template literals.
- Access all the values and keys of an object.
- Use `find()`, `filter()`, and `map()` to solve different problems.
- Use `sort()` to sort arrays in various ways.

Project setup

Download the Qualified assessment files to your computer.

Then `cd` into the assessment folder.

Next, install dependencies locally by running this command:

```
npm i
```

To run the tests, you can run the following command:

```
npm test
```

To watch how the code you write affects the application website, you can run the following command. This command will start a server and take over your terminal window. To stop the server from running, you can press `Control+C`.

```
npm start
```

Existing Files

Folder/file path	Description
`public/accounts.html`	The HTML document that holds the accounts page. You **don't** need to modify this file for the tests to pass.
`public/books.html`	The HTML document that holds the books page. You **don't** need to modify this file for the tests to pass.

Folder/file path	Description
`public/index.html`	The HTML document that holds the main library page. You **don't** need to modify this file for the tests to pass.
`public/data`	Contains the accounts, authors, and books data for the library project. You **don't** need to modify any of the files within this folder for the tests to pass.
`public/renderers`	Contains the scripts for rendering the accounts, authors, and books templates. You **don't** need to modify any of the files within this folder for the tests to pass.
`public/src`	Contains the scripts for the accounts, authors, and books pages. You **will** need to complete the functions inside these scripts for the tests to pass.
`test`	This folder holds the unit tests of the project. You **don't** need to modify any of the files within this folder.

Instructions

You are tasked with building a number of different algorithms that will help complete an administrative site for a local library. All of the functions will work on three common datasets. The datasets are related, and at times, you will need to work with multiple datasets to solve the problem at hand.

While working on these problems, you have both the tests and the live site to act as a guide. You can solve the tasks in any order and are encouraged to organize your code how you like.

All of the files you need to edit are inside of the `public/src/` directory.

While working on this project you *should*:

- Use well-named variables, in particular avoiding any single-letter variables that lack meaning.
- Use native array methods like `find()`, `filter()`, and `map()`.
- Make use of both function declarations and arrow functions.
- Make use of at least one of the following JavaScript features: ternary operators, the spread operator, object shorthand, array and object destructuring, and `for/in` loops.

While working on this project you *should not:*

- Change the names of the functions.
- Edit any of the files outside of the `public/src/` directory.

If you feel as though one of your solutions is working but something isn't showing up right on the site or in the tests, reach out for help.

Datasets

There are three datasets that are a part of this project: `accounts`, `authors`, and `books`.

Accounts

You can view all of the accounts data inside of the `public/data/` directory. Each account is an object with the following shape:

```
{
  "id": "5f446f2ecfaf0310387c9603",
  "name": {
    "first": "Esther",
    "last": "Tucker"
  },
  "picture": "https://api.adorable.io/avatars/75/esther.tucker@zillacon.me",
  "age": 25,
  "company": "ZILLACON",
  "email": "esther.tucker@zillacon.me",
  "registered": "Thursday, May 28, 2015 2:51 PM"
}
```

An account represents a person who is registered with the library. Accounts can take out and return books.

Authors

You can view all of the authors data inside of the `public/data/` directory. Each author is an object with the following shape:

```
{
  "id": 0,
  "name": {
    "first": "Lucia",
    "last": "Moreno"
  }
}
```

An author represents someone who wrote one or more books in the library.

Note: Author's IDs are set to be numbers, whereas the other two datasets use string IDs.

Books

You can view all of the books data inside of the `public/data/` directory. Each book is an object with the following shape:

```
{
  "id": "5f4471327864ee880caf5afc",
  "title": "reprehenderit quis laboris adipisicing et",
  "genre": "Poetry",
  "authorId": 20,
  "borrows": [
    {
      "id": "5f446f2e2a4fcd687493a775",
      "returned": false
    },
    {
      "id": "5f446f2ebe8314bcec531cc5",
      "returned": true
    },
    {
      "id": "5f446f2ea508b6a99c3e42c6",
      "returned": true
    }
  ]
}
```

Each book represents a physical book but also contains additional information. In particular:

- The `authorId` matches up with an author. It represents who wrote the book.
- The `borrows` array is a list of transactions that have occurred with this book. For example, the above book has been borrowed three times.
    - The `id` for each "borrow" record matches with an account ID. In the above example, the account with an ID of `"5f446f2e2a4fcd687493a775"` has not yet returned the book, meaning they still are in possession of it.

Functions

You are tasked with writing several functions that work with the above datasets. The instructions for the functions can be found in the `docs/` folder.

Note: In addition to needing to pass the tests and requirements in the instructions here, please review the Rubric Requirements for the human-graded part of this project in your Thinkful curriculum page.