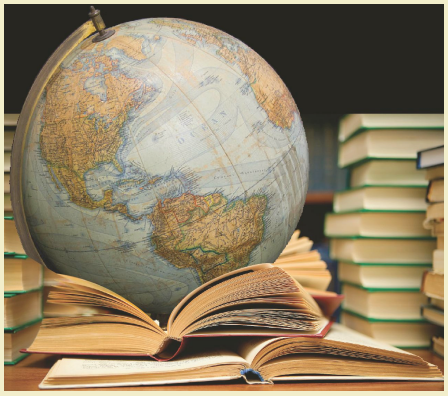
BookRUs

Technical Report: Phase 4



Matthew Escobar, Francisco Reyna, William Eng,
Hrithik Ramganesh, Elliot Sims

Motivation

“That’s the thing about books. They let you travel without moving your feet.”

- American author Jhumpa Lahiri

Our mission is to emphasize the literary contributions of different cultures around the world. BookRUs was designed to include information regarding famous publications around the globe, their authors, and the countries they hail from. This allows us to connect and spread knowledge from distant lifestyles and faraway populations to others through inspiring stories, insightful biographies, and cross-country cultural exchange. Therefore, we can create a more mindful atmosphere as people across the globe become more conscious and appreciative of each other!

As of now, the vision of our website incorporates rich backgrounds, summaries, descriptions, and images from authors, books, and countries alike to resemble a digitized library where anyone can digest the cultural significance of a piece of literature beyond what they read on the page. Additionally, we hope that our website will aid readers in their thirst for knowledge by helping connect them to novel books, authors, and cultures that pique their interests.

User Stories

(Phase 1) Our Stories to Find-a-Home:

- **Immigration Rules**
  - “I would like to know what immigration to each country is like. The process of traveling to a country and getting approved to stay in said country may be important if I want to choose somewhere to stay temporarily. It also lets me determine whether one country is more difficult to stay in than another, despite other factors such as employment rate, cost of living, etc.”

- **Neighboring Countries**
  - “In light of recent events, I would like to see a short "History" section that maybe outlines the creation of the selected country and explains its relations with neighboring countries. For example, the fact that there is open travel between EU countries is a positive attribute. On the other hand, maybe the country has tense diplomatic relations with someone nearby, which could incite conflict in the future. This would be great for long-term planning and is a way an asylee can make sure that they aren't escaping into another potential conflict.”

- **Fellow Immigrants**
    - “I would like to know how many people are also immigrants maybe from the same country as I am that live relatively close by or even at work. Is there like a small community that I can join when I move in and come in to work. Maybe I could get their contact info if they were relocated by the same service as I did for example findahome.”

- **Data Format**
    - “As someone who wears contacts/glasses, it hurts my eyes to read the text data when there is no spacing between it and the side of the page. Is there a way to pad the text to make it more readable? Also, is there a way to style the text more to make it more accessible beyond the padding? Such as bolding and organizing sections?”

- **Safety Net Policy**
    - “Suppose I get kicked out of my job. Will I be helped in finding a new job or will I need to be relocated to find a job. Will I keep the same occupation or will it be changed and salary.”

(Phase 1) GerryMap's Customer Stories to Us:

- **Information about the length of the book**
    - "If possible, I would like to see a few more book attributes. One of the main factors I look at when finding a book is the book's length. Is it possible to find data on the number of pages in the book? With book-length information, one can have one extra data point to decide if the book is for them."
    - Resolution: Yes, it's possible to find data on the number of pages in the book! We think it's a great suggestion and have thus added it to our site.

- **Information about book pricing/availability**
    - "If I looked through the website and found a book that I really liked and wanted to purchase to support the author, it'd be nice to see pricing information or places to buy the book from various retailers. Sometimes, when I try to find books from other countries or cultures, they can be somewhat hard to find. A central place to find information about this would be really handy!"
    - Resolution: Finding the pricing data will take a little extra work in the next phase, but it is doable. While the site focuses on the culture surrounding the books, authors, and countries, we think it's a great idea to provide clients the opportunity to dig deeper by prospecting books to purchase if they so choose.

- **Information about the book's reception**
    - "An important part of learning about a book's relationship with a given culture is to examine its reception. An interesting piece of data to add would be some reviews that the book has received. This would add context to how the book fits in with the culture that the author originates from, or how readers from another culture might interpret the book."
    - Resolution: Weighing the reception of the book is a great idea! This will likely be implemented in another phase as it does require some extra consideration: where will the ratings come from? How does the region where the ratings come from affect its reception? *Etc*.

- **History of the country**
    - "For the culture of the countries, I would like to see the history of the country and the authors in the country can be grouped by different eras. For example, the eras can be World War I and Great Depression. This will let me know more about the background of the author's life. Moreover, I can see if the same era influences the authors to create similar genres."
    - Resolution: While this is an interesting idea, it's regrettable to say that we don't see this as a feasible feature to implement given the resources made available to us. The further back we go, the less we

know of the world. Narrowing down that already limited knowledge to a specific region and a specific person is a challenge that we don't believe can be encompassed within the scope of this project.

- **Region attribute**
  - "Since countries and cultures are organized geographically around the world, it would make sense to introduce more geographic attributes to the data on the site. For example, the United States and Canada are probably more culturally similar than the United States and China. It would be helpful to group countries and cultures by region, to be able to see how cultures within the same region might be similar or different based on the books that are produced."
  - Resolution: Great idea! Geography definitely has an influence on the dispersion of culture and thus we have added it to our site.

(Phase 2) Our Stories to Find-a-Home:

- **Documentation**
  - “I am a customer who likes building websites. Is it possible to add a link to your source code and API documentation for FindaHome? I want to learn more about the structure of your website and use the country API to get data for my own websites.”

- **News Cards Language**
  - “Hi, I am a customer who has trouble identifying languages. I want to translate your news cards, but I don't see an attribute for their language. I have to click on the card to go to the instance page just to find the language. It would help a ton if I could find it more easily, so I can translate the news cards to find which one I want before clicking on it.”

- **Visual Appeal**
  - “Overall I feel like the pages are a bit abrasive to view. The text on each page feels super close to the left side of the webpage and the numbers sort of swirl together as I read through the page. I think the pages would be more visually appealing if you can separate out the data into rows/columns and maybe even add data visuals (pie charts/bars for percentages, life satisfaction (out of ten?), etc).”

- **Currency**
    - “As a person who likes to see the different kinds of charities available, I am confused at what kind of currency I am looking at when viewing the funding of these charities. Perhaps adding a currency (i.e. USD, EUR, KRW, etc.) or standardizing all funding values with USD (and stating such) would be helpful when quickly looking at the valuation of charities.”

- **Groups**
    - “Suppose I was with a family or a group of people that wanted to find a home together or separate but within a reasonable distance. Would your application be sufficient for my needs? I would like to see a more local scope of the homes such as a neighborhood in the city.”

(Phase 2) GerryMap's Customer Stories to Us:

- **Desiring More Author Information**
    - "I would like to know more information about each author. Here are examples of data points that you could add: death of authors, book sales data, famous quotes, and era. If possible, one should be able to filter or search based off these attributes."
    - Resolution: This is a great suggestion! We've added more information about each author in this phase, although that doesn't include everything you suggested. Book sales, quotes, era will have to be gathered outside of the APIs we are currently using. I hesitate to say that more author information is outside of the scope of this project because some of the attributes you list warrant some more web hunting. We'll add what we can find!

- **UI/UX enhancement**
    - "As a user who cares about web layout, I would like to see the page layout shows the importance of the field. For example, for author pages, the image should show on top, and the book genre shouldn't be on the same line as gender and birth date. I like the design of the splash page, which makes me feel like walking into a bookstore, however, I suggest making "Books", "Authors", and "Countries" all hyperlinks at the bottom of the splash page."
    - Resolution: Great suggestion! We should be able to implement these enhancements this phase.

- **Country-Specific Literary Information**
    - ”I’d like to see more information about each country’s literary traditions or their most influential works on their model pages. I think it’d help tie the country models more closely into the website theme. Highlighting each country’s most influential books would also help integrate the books models into the country models.”
    - Resolution: You're correct that the most influential books per country will give insight into literary traditions and integrate both the book and country models! However, measuring influence is beyond the scope of this project. If it's any consolation, authors will be mapped to their respective countries and their most influential works can be accessed through the books that they have written.

- **Model Descriptions**
    - “When reading the model descriptions, there are a lot of brackets with numbers, which I assume are citations parsed from Wikipedia. However, these make for a poorly-flowing reading experience when I am trying to learn about each country or author. I’d like these descriptions to be better parsed/processed to remove any trace of the fact that they were parsed from elsewhere.”
    - Resolution: Yes, you're correct that those are citations from Wikipedia. We will remove those citations in this phase.

- **Download Readings**
    - “I can see that many of the works available on the site will be old classics like Shakespeare and George Orwell. Many of these books are widely available for free online. If possible, I would like to see download links for stories that are offered digitally.”
    - Resolution: In our example version of the website, yes, a lot of the old classics are available and free online. There are, however, a lot of publications that will be included that don't share those same properties. Although your suggestion is great, finding a way to download all of these readings is beyond the scope of this project.

(Phase 3) Our Stories to Find-a-Home:

- **Clicking Tables**
    - “Hi, as a customer, I had trouble realizing that the tables were clickable to take me to the instance pages. I like the highlight effect that indicates there is some action with the table. However, I think changing the cursor to a pointer might help other customers realize we can click on the table.”

- **Homepage Extra Space**
    - “Hi, I pulled up your website, and it looks beautiful. However, I noticed that the splash art stays in the same spot regardless of changing my window size. If possible, I think it would improve the look of the home page to make the spacing more dynamic.”

- **News/Charity Link Windows**
    - “I think it would be nice to have a widget/window where the user can click the left or the right side of the widget to view different news articles or charities on the instance pages instead of having a list of hyperlinks. The window can display the name of the article placed upon an image.”

- **Translated News Articles**
    - “Is it possible to allow the news to be translated to like one common language. Or it could be translated to any language preferred through maybe google translate or

other methods? It would be really helpful otherwise I would be forced to search for the same news but in my language."

- **Visibility to Numbers**
    - "Browsing your Countries model page, I see some very large numbers for your population and sometimes household net worth. I think it would be nice if you could add commas to the larger numbers that you display (i.e. funds raised/goals for charities, country population). I think this would enhance the view for someone like myself, and make it easier to grasp the scale (millions vs billions, tens vs hundreds of thousands, etc.)."

(Phase 3) GerryMap's Customer Stories to Us:

- **Improve Data Formatting**
    - "I would like to have the data for each instance of countries. For example, population should be formatted with commas instead of just an integer string. Likewise, Latitude and Longitude coordinates should include N°, W°, S°, and E°."
    - Resolution: On the instance pages, the coordinates are no longer listed--instead there's a map! The population will also be displayed with commas.

- **Information on Best Work**
    - "I noticed that each author has a "best work" attribute. As a user, it isn't inherent what this attribute might mean. I would love to see some information on what that entails, such as a section on the About page explaining whether this refers to the authors' most popular works, or their most highly rated works, or something else."
    - Resolution: While your suggestion is sensible, it's with regret to say that, with the resources that we are using, the author's information on the best work isn't always readily available--even as an instance page. As of now the best work will remain as it is but perhaps in the future we'll be able to add more information on the authors' best works.

- **Better Clarity on Book Language**

    - “It looks like currently, the book’s language information just uses the language code - for example, “en” presumably refers to English. However, this could be unclear for more obscure languages. Changing this to use the actual language name would be more clear and also give a more polished appearance to the website.”
    - Resolution: Great suggestion, we'll implement this into our site!

- **Navigating the Site**
    - “As a user, I find it more intuitive when text is clickable rather than images. I found it hard to navigate the country model page, for example, since I didn’t immediately realize that the flags were links to other pages. I think in these cases, both the names and the images (or even the whole card) should be clickable to make it more obvious to new users that each instance has its own page.”
    - Resolution: The entire cards will be clickable and are sensitive to the mouse hovering over the card. Hopefully this makes site navigation more intuitive!

- **Country Model Page Display**
    - “As a user who wants to find Germany from the country list, I wish the country can show alphabetically so that I can quickly find it. Now it shows between the Czech Republic and Djibouti. Moreover, I wish to see more attributes shown under the country, such as how many books or authors are in this country.”

- Resolution: Thank you for the suggestion! We will make sure you are able to view the countries alphabetically and search countries by name. The number of authors will be visible on the instance page but we will revisit the attributes displayed on the model page.

RESTful API

Our Postman API documentation can be found here. (https://documenter.getpostman.com/view/19701903/UVkvJYLx) As of right now, the documentation for RESTful API for BookRUs contains six basic GET requests for the three of our models. Each model has two requests: one which retrieves a page of instances of the model and one which retrieves one instance of the model given an ID. With the API endpoint of https://api.bookrus.me, our calls for each model look like the following:

1. https://api.bookrus.me/books

   Retrieves all books and takes 8 parameters that affect filtering/searching/sorting of the result:

   - Limit: Takes an integer. Limits how many instances are retrieved per page
   - Sort: Takes a string. Determines what model attribute to sort by.
   - Search: Takes a string. Used to search for book titles with the specified string. Spaces are delimited.
   - Page: Takes an integer. Determines what page to view.
   - Language: Takes a string. Filter by language.
   - Length: Takes two integers separated by a hyphen. Filters for books with page counts between the two integers (inclusive).
   - Maturity: Takes a string. Filters by maturity.

- Genre: Takes a string. Filters by genre.

2. https://api.bookrus.me/books/<id>
   Takes an <id> in the form of an integer. Retrieves a single instance of a book.
3. https://api.bookrus.me/authors
   Retrieves all authors and takes 7 parameters that affect filtering/searching/sorting of the result:
   - Limit: Takes an integer. Limits how many instances are retrieved per page
   - Sort: Takes a string. Determines what model attribute to sort by.
   - Search: Takes a string. Used to search for author names with the specified string. Spaces are delimited.
   - Page: Takes an integer. Determines what page to view.
   - Nationality: Takes a string. Filters by nationality.
   - Main-genre: Takes a string. Filters by author's main genre.
   - Work-count: Takes two integers separated by a hyphen. Filters for authors with work counts between the two integers (inclusive).
4. https://api.bookrus.me/authors/<id>
   Takes an <id> in the form of an integer. Retrieves a single instance of an author.
5. https://api.bookrus.me/countries
   Retrieves all countries and takes 6 parameters that affect filtering/searching/sorting of the result:

- Limit: Takes an integer. Limits how many instances are retrieved per page
- Sort: Takes a string. Determines what model attribute to sort by.
- Search: Takes a string. Used to search for country names with the specified string. Spaces are delimited.
- Page: Takes an integer. Determines what page to view.
- Region: Takes a string. Filters by region.
- Population: Takes two integers separated by a hyphen. Filters for countries with work counts between the two integers (inclusive).

6. https://api.bookrus.me/countries/<id>
   Takes an <id> in the form of an integer. Retrieves a single instance of a country.

Upon a successful request, each model's instance(s) will be listed out and the user will be able to access all attributes of each instance (i.e. book author, country code, author bio, *etc*.) The list of attributes, the available requests, and the actual practicality of the API will grow as we continue to develop the site.

Models

Presently, our project consists of three models. Data for each model comes from the Google Books API, Wikimedia, Wikipedia, World Bank Country/Region API, and the Open Library API.

We chose to make all 5 main attributes of each model searchable and sortable (no filterable).

Books

- The books make up the literary component of our project.
- Books have a one to one relationship with authors and countries; there is only one author per book and one country per book.
- Sortable/Sortable attributes: Title, Author, Language, Genre, Length
- Searchable attributes: Title, Author, Language, Genre, Length
- Other attributes: Maturity, Date Published
- Media: Synopsis, Image, Map

Authors

- The authors serve as a connection between their literary works and both their personal culture and the culture of their nationality.
- Authors have a one to one relationship with countries; there is only one country per author. Meanwhile, authors have a

one to many relationship with books; there can be many books per author.
- Sortable/Filterable attributes: Name, Best Work, Work Count, Main Genre, Nationality
- Searchable attributes: Name, Best Work, Work Count, Main Genre, Nationality
- Other attributes: Born, Died
- Media: Bio, Image, Map

Countries

- The countries are the macroscopic units of the culture of our project.
- Countries have a one to many relationship with books and authors; there can be many books and authors per country.
- Sortable/Filterable attributes: Name, Region, Population, Latitude, Longitude
- Searchable attributes: Name, Region, Population, Latitude, Longitude
- Other attributes: Capital, Demonym, Total Authors, Languages
- Media: Description, Image, Map

Data Sources

- Google Book API
  - https://developers.google.com/books/docs/overview
  - Gives book info and images
- World Bank Country/Region API
  - https://datahelpdesk.worldbank.org/knowledgebase/articles/898590-country-api-queries
  - Gives country info including region
- Open Library API
  - https://openlibrary.org/developers/api
  - Gives book and author info
- Wiki API
  - https://www.mediawiki.org/wiki/API:Main_page
  - Multiple Wikipedia APIs to get long descriptions of objects

Tools

Frontend

- **React:** A JavaScript library for building user interfaces for single-page applications. We used it to create the frontend of our app. React is popular due to its Component feature that acts as building blocks and saves significant time.
- **React-Bootstrap:** A component-based library that provides Bootstrap components as React components. We used it to easily make our navbar and design each of our site pages.
- **React-Router:** A React library that enables navigation among views of various components. We used it to navigate between pages.
- **Jest:** Unit testing software used to test our frontend React components to ensure all components would render without errors.
- **Selenium:** GUI unit testing software used to test our frontend web page from the customer point of view and ensure page components work as intended for the user. This includes testing button navigation and hyperlinks to other pages.
- **Prettier:** An opinionated code formatter that works on most files. We ran it before our linter to fix most style issues.
- **Eslint/Eslint-Config-Airbnb:** A linter for JavaScript that is used for most of our frontend work. The Airbnb config makes the linter follow Airbnb style guidelines, which is the most popular guide for React apps.

- **AWS Amplify:** Amazon Web Services' streamlined hosting service to make hosting clients' sites easy. This is what we used to host our website.
- **Rechart:** React component library used to make charts/graphs on data visualization pages.

Backend

- **Flask:** Flask is a Python-based web application framework that we used to deploy our front-facing API application, as well as process and return requests to the database.
- **Flask-Restless-NG:** A Flask plugin that we utilized to create the API endpoint blueprints, which Flask would turn into a functioning web application. This plugin was removed in Phase III in favor of a manual API endpoint implementation to allow searching/filtering/sorting.
- **Flask-CORS:** A Flask plugin which we used to add cross origin headers to our backend responses, allowing interaction between our frontend and backend.
- **Flask-SQLAlchemy:** A Flask plugin we used to read in our database models into Flask, so that our API is able to make queries with knowledge of our database table structure.
- **SQLAlchemy:** A library that we used to interact with our production database, allowing us to make queries and input data into our database using Python.
- **unittest:** Library used for building and running our backend python unit tests. We combined this with Flask's test client to implement unit testing on our backend endpoints locally.

- **Postman:** An application for testing and documenting APIs. In this phase, we used it to complete documentation of the API and create API tests that will run through GitLab.
- **Elastic Beanstalk:** An AWS service that allows us to host our backend database and API.
- **Black:** An opinionated formatter which we used to format our backend python files.

Other

- **GitLab:** A version control system that facilitates project collaboration, management, and development. We used this to manage the workflow of our project and to link to AWS Amplify and Elastic Beanstalk. We also used their CI/CD pipeline to continuously perform unit testing and check for issues during development.
- **Discord:** A communication platform. This is what we used to communicate and coordinate as a team whether it's through calls or messaging.
- **Docker:** Docker is a container system that we used to test and deploy both frontend and backend systems with their respective required images containing the tools seen above.
- **Namecheap:** A domain name provider and register. We used this to find a free domain name that would replace the domain given to use by AWS.

Hosting

To host our frontend site, we used AWS Amplify and Namecheap. We started by purchasing the domain name “bookrus.me” on Namecheap for free since Namecheap has special offers for students. Next, we set up AWS Amplify which was, as advertised, quick and easy to use. We linked AWS Amplify up to our GitLab’s main branch frontend and dev branch frontend. The main difficulty of the process resulted from trying to route traffic reaching bookrus.me to the site AWS was hosting for us. For the sake of recording a lesson, we did as follows:

1. Added the CNAME record AWS provided into Namecheap
2. Added the second CNAME record AWS provided into Namecheap
3. Remove all A records from Namecheap
4. Added the ALIAS record AWS provided into Namecheap

Once we configured everything correctly in Namecheap, AWS Amplify took care of the rest. We also took the precaution of disabling automatic build updates upon committing to the dev branch to save the amount of build hours allotted to us by AWS’s free tier.

To host our API, we used AWS Elastic Beanstalk, Amazon Relational Database Services, and Namecheap. The first step we took was hosting our backend on Elastic Beanstalk using the Elastic Beanstalk CLI, Docker, nginx, and uwsgi. Once we were hosted, we created a HTTP and an HTTPS load balancer and

configured a domain name to which we hooked up to Namecheap via a CNAME record. Next, we created a publicly accessible relational database on Amazon RDS with PostgreSQL (the free tier of course). Then, using the endpoint on our database, we access it through pgAdmin4 to help us visualize our database. The tedious part of this set up is that we opted out of most automatic deployments. So, to update our database, we need to take a snapshot via AWS RDS, decouple our last database snapshot from Elastic Beanstalk, then finally couple the latest snapshot on Elastic Beanstalk. Additionally, Elastic Beanstalk needs to redeploy each time there's a change to the backend's app.py file. It's a decent amount of overhead that demands a bit of patience.

Phase II Features

- **Pagination:** We implemented pagination using React Bootstrap's Pagination component as a foundation for our own MyPagination component. In MyPagination, we take the currentPage and totalInstances (of the model) as props. These two integers alone allow us to calculate total pages, range of instances per page, and design onClick functions to update the currentPage depending on where the customer wants to go (first, prev, next, end). We initialize currentPage to the first page and changing it will update our website to the new page. Also, we restrict customers from going to pages out-of-range by disabling the Pagination buttons to those pages and added a dropdown button for customers to change the number of results per page. We pass currentPage as "page" and numResults as "limit" to in our API url params. Then, in the backend, we pass those args to their respective query functions, so the data is paginated correctly.

- **Database:** Our database is hosted under Amazon Relational Database Services (RDS) using as our engine PostgreSQL (PSQL) and pgAdmin4. To keep things as cheap as possible, we selected RDS's free tier and about a fifth of the allocated memory. We also set it up to be publicly accessible, but a password is required to edit the database itself. After editing inbound rules on RDS to allow for inbound connections, we

use the credentials we set up in Amazon RDS to log into the database via pgAdmin4 to easily manipulate our data, create tables, et cetera through PSQL. To scrape data into our database, we use the following packages:

1. Psycopg2: To connect to the database with python source code
2. Flask: To initialize our app
3. SQLAlchemy: To make our database instance

From there, we're able to edit the database instance of SQLAlchemy as easily as we can edit any python class. Using urllib and json, we access our APIs and pass values into our database model objects (what would become rows in the database) and add each one to our database session. Once we're finished, we commit the database at the end of our script. However, since we opted to redeploy AWS manually, every edit to the database must be updated through RDS. To do this, we use RDS to take a snapshot of our database, and then we decouple our old snapshot from our app on Elastic Beanstalk and sub in our newest snapshot of the database.

Phase III Features

- **Sorting/Searching/Filtering**
  - We chose to make all 5 main attributes of each model both searchable and sortable. We did not implement filterable attributes, because we thought the UI would be too cluttered and searchable is a better option to filterable. It also allows us to not worry about the user searching over filtered data.
  - **Backend:** During Phase II we learned that Flask-Restless-NG, the plugin that enabled our API in Flask, had issues that would prevent us from being able to complete a stable sorting/searching system. As a result, this phase we rewrote the backend to perform the same operations as Flask-Restless-NG using only Flask and Flask-SQLAlchemy, which gives us much greater flexibility to modify our database queries, change our API output, and implement sorting/searching/filtering.
    For our implementation of sorting/searching, we implemented query parameters on our 3 models:
    - search - takes in phrases, separates each phrase and does a search on searchable attributes using SQL parameter “ilike” for strings (i.e. book authors or book languages) or equivalency searches for numericals, such as author work count, filtering

these results and passing only those results to the user.
    - sort - For sorting, we can pass a “sort” API parameter that will order the paginated results by the attribute requested, implemented by using Python’s getattr() to find the requested attribute in the model and Flask_SQLAlchemy’s order_by() to order by said attribute.
- **Frontend:**
    - **Model Specific:** On the model pages, we added a new NavBar to hold all the UI components for searching, sorting, and paginating. Therefore, on the NavBar, we have a dropdown for the user to choose which attribute of the model they wish to sort by, and we have a search bar. The pagination component was added to the far side of the navbar. When the user selects a sort option, we pass a “sort” param through our API url, and similarly when the user types in a search, we pass a “search” param. We explained previously how we use those values in the backend. For additional features, we added a “No Sort” option that the user can select in the dropdown to remove sorting and the user is allowed to search and sort simultaneously. Since we did not implement filtering, all searching is done over the full data range.
    - **Global:** We created a new search result page. It has a similar new NavBar minus the sorting and

pagination components. Since it is a global search, the search bar will search across the main 5 attributes of all 3 models. This is done by simply combining the 3 API calls we used for our model specific searches. We then designed a tabbed component that treats the 3 model searches as independent tables with their own unique paginations. You can think of the global search result page being 3 embedded viewers of each model specific search. Since we are not designing a search result algorithm that returns “most clicks” or “most similar,” our design allows us to organize the random data for the user. If the user wants to perform sorts or more searches on a specific model, he/she can use the website NavBar to travel to the model specific searches on the model pages. Lastly, we display a total results count for the user.

Phase IV Features

- **Visualizations**
  - Our main tool for the visualizations was Rechart. It is a library of custom React components to build simple charts/graphs. For both our visualizations and our provider's, we used the same bar chart, pie chart, and radar chart. Before creating the charts/graphs, we made API calls to the respective databases, and then we organized the data to be inserted into the charts/graphs.
  - We noticed a significant time difference between a few API calls and multiple. Therefore, we tried to prioritize using a couple big API calls to grab all the data and then organizing with Javascript instead of sending sorting/filtering params through the API urls.
  - Lastly, our biggest challenge was AWS's limits on API calls. Our customer group overloaded our backend server when they made an API call to our book database of 40k instances. We managed to fix it pretty quickly, however, we did the exact same thing and overloaded our provider's backend server. Unfortunately, our providers did not fix their backend for over a week and until the night before the due date (our help attempts also failed). Therefore, we restricted our API calls to small model databases and/or used pagination to prevent server overload. This also gave us

a small time frame to test things with their API, so our provider's visualization page has a few issues such as a slow load and/or no spinner.