EE 461L Term Project Report

Team : The Bookish People
Project : Booklopedia
Canvas Group : morning-2
Github Repository : https://github.com/VishruthiR/EE461L_IDB
Website Link: http://booklopedia.appspot.com/

Meet the Team :

Kumaran Arulmani
Email: kumaranarulmani@utexas.edu
Github Username: @kumosumo

Vishruthi Ramaswamy
Email: vishruthi.ramaswamy@utexas.edu
Github Username: @vishruthir

Jaino Vennatt
Email: vennatt.jaino@gmail.com
Github Username: @jainovennatt
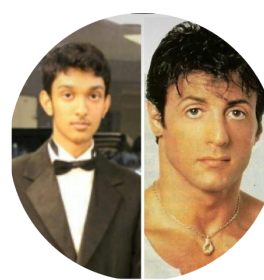

David Day
Email: david.day@utexas.edu
Github Username: @davidday99

Matthew Jiang
Email: jiang.matthew22@gmail.com
Github Username: @myj99

Siddhartha Shetkar
Email: sshetkar@gmail.com
Github Username: @sshetkar3858

Motivation :

Our motivation for making this website was to combine a lot of our favorite book models from different websites (google, amazon, barnes & noble) and turn it into one cohesive website to find all the information. We also want to implement a user profile for the website where users can personalize their experience to fit their interests.

Users :

We expect our primary users to be online book lovers (users that would usually engage with Google Books or Amazon Books). Primarily, we want to target book lovers who want to be a part of an online community to share their interests in the form of personal favorites and personal recommendations.

Phase Leads :

Phase I   - Matthew Jiang
Phase II   - Kumaran Arulmani
Phase III - Siddhartha Shetkar

User Stories :

Name :   **1**
Genre Description
Assignee :
Siddhartha Shetkar
Description :
As a user, I want to be able to see descriptions of various genres and so I can further explore different areas of literature.
Development Process :
Create descriptions to put in a database with the corresponding type of genre. On the genre page display the description.
Estimated Time to Completion :
2 hours
Actual Time to Completion :
2 hours

Name :   **2**

Editor's Choice Carousel

Assignee :

David Day

Description :

As a user, I want to be able to see the "Editor's Choice" of books by a favorite author or of a certain genre so I can further explore books that may fit my interest and likings.

Development Process :

Create a carousel of images of books pulled from a database by the same author (for author page), or classified as the same genre (for genre page).

Estimated Time to Completion :

2 hours

Actual Time to Completion :

2 hours

Name :   **3**

Book Summary

Assignee :

Vishruthi Ramaswamy

Description :

As a user, I want to be able to read short descriptions (blurbs) of books so I can get a sense of what the book is about and gage my interest.

Development Process :

Using API store data on book summaries in the database. Display corresponding summary to book on page.

Estimated Time to Completion :

2 hours

Actual Time to Completion :

2 hours

Name :   **4**

Book Images

Assignee :

Jaino Vennatt

Description :

As a user, I want to be able to visualize different books that I am searching, so when I am looking to purchase a book I have an idea of what the book looks like.

Development Process :

Store images from API of books. Display book images on corresponding pages.

Estimated Time to Completion :

2 hours

Actual Time to Completion :

2 hours

Name :   **5**

Navigation Bar

Assignee :

Matthew Jiang

Description :

As a user, I want to be able to easily access the different pages the site has to offer so I can navigate through the website with ease and find information quickly.

Development Process :

Have links to different pages on a side bar available on all pages.

Estimated Time to Completion :

2 hours

Actual Time to Completion :

2 hours

Name :   **6**

About Page

Assignee :

Kumaran Arulmani

Description :

As a developer, I want to be able to see the different contributions that went into the website and GitHub statistics on interactions/commits with the project so I can better understand everyone's contribution.

Development Process :

Have a dynamic webpage pulling data from the GitHub regarding each team member's profile (picture, biography, commits, interactions) and overall project statistics.

Estimated Time to Completion :

2 hours

Actual Time to Completion :

2 hours

Name :   **7**

Search Functionality

Assignee :

Kumaran Arulmani

Description :

As a user, I want to be able to search by Book Title, Author, Genre and Publisher so I can further explore literary works related to one another.

Development Process :

Have search (input) fields for each model, create a query based on input and display the results from the database onto the Results page (with Pagination functionality).

Estimated Time to Completion :

15 hours

Actual Time to Completion :

20 hours

Name :   **8**

Accessibility & Design

Assignee :

Vishruthi Ramaswamy

Description :

As a user, I want to be able to view information easily and quickly, and the website should be inviting to use.

Development Process :

Use Material UI along with React to create an aesthetically pleasing and easy to use website design. Elements should be easy to read and can traverse through the website with no difficulty.

Estimated Time to Completion :

17 hours

Actual Time to Completion :

19 hours

Name :   **9**

Pagination for Search Results

Assignee :

Jaino Vennatt

Description :

As a user, I want to be able to view my search results in an organized manner (only showing 10 results per page) and sift through the pages to be able to read information quickly and easily.

Development Process :

Use Material UI along with React to create a pagination bar (using the template provided by Material UI). Each linked results page should send a new query to the database to pull corresponding results of the query (Page 1: Results 0-9, Page 2 : Result 10-19).

Estimated Time to Completion :

12 hours

Actual Time to Completion :

15 hours

Name :   **10**

Dynamically Created Model Pages

Assignees :

David Day & Sid Shetkar

Description :

As a user, I want to be able to access a page for every search result (if I search for a book, I am able to click on the book to be redirected to a page with more information - same for other models) to further organize and get more information about each search result.

Development Process :

Create a general template for each model page (Book, Genre, Author) using Material UI & React, make the parameters for information variables that can be passed in from a query (for example: If I click on the Book Title of "The Great Gatsby" the corresponding author, ISBN, book summary, etc. should populate on the Book page).

Estimated Time to Completion :

15 hours

Actual Time to Completion :

17 hours

Name :   **11**

Navigation Bar (Part 2)

Assignee :

Matthew Jiang

Description :

As a user, I want to be able to easily access the different pages the site has to offer so I can navigate through the website with ease and find information quickly.

Development Process :

Have links to different pages on a side bar available on all pages. Recreate using Material UI & React, also add search functionality to Navigation Bar. Should be available on all pages.

Estimated Time to Completion :

5 hours

Actual Time to Completion :

7 hours

Name : **12**

Genre of Recommended Books

Assignee :

Kumaran Arulmani

Description :

As a user, I would like to know what genre a book is in, in addition to knowing the author to be able to explore similar books in the same genre.

Development Process :

Make sure when querying the information, the genre of book is also available from the database.

Estimated Time to Completion :

5 hours

Actual Time to Completion :

5 hours

Name : **13**

Book Number of Pages

Assignee :

Jaino Vennatt

Description :

As a user, I would like to know the number of pages a book encompasses to help me better gage an estimated reading time.

Development Process :

Make sure when putting information into the database, the book's number of pages is also pulled.

Estimated Time to Completion :

2 hours

Actual Time to Completion :

2 hours

Name :   **14**

Loading Icon

Assignee :

Matthew Jiang

Description :

As a user, I would like the website to have a loading icon that is displayed when pages are fetching the information so it is aesthetically pleasing.

Development Process :

Using react, create a loading icon for when pages are trying to load information.

Estimated Time to Completion :

2 hours

Actual Time to Completion :

2 hours

Name :   **15**

Filtering

Assignee :

Kumaran Arulmani, Sid Shetkar

Description :

As a user, I would like to have buttons available to filter my search results by - Books, Genres, & Authors.

Development Process :

When querying the database, return search results only corresponding to filter preference.

Estimated Time to Completion :

6 hours

Actual Time to Completion :

6 hours

Name : **16**

Book Images

Assignee :

Vishruthi Ramaswamy

Description :

As a user, when I am looking at recommended books, or a specific book page I would like to view the images of books so I can get a better idea of what book I am reading information about.

Development Process :

When querying the database for the recommended books carousel or specific pages, make sure to display the book image in the header.

Estimated Time to Completion :

4 hours

Actual Time to Completion :

4 hours

Name : **17**

Visual Indicators (Search Results)

Assignee :

David Day

Description :

As a user, I would like to have visual indicators on search results, so I can more easily read/see information about results.

Development Process :

When querying the database for search results, make sure to display results with images to further give the users visual aids.

Estimated Time to Completion :

4 hours

Actual Time to Completion :

4 hours

Name :   **18**

Linked Models

Assignee :

Siddhartha Shetkar

Description :

As a user, I would like to have links to other model Pages from specified model pages to learn more information (Book would have links to Author & Genre of Book).

Development Process :

Make links to Authors/Genres/Books specified by generated another query whenever the user clicks on a link, that will populate the specified page with the correct information.

Estimated Time to Completion :

5 hours

Actual Time to Completion :

5 hours

Name :   **19**

Homepage Navigation

Assignee :

Matthew Jiang

Description :

As a user, I would like to have quick access to the homepage through a navigation bar of sorts, to easily navigate through the website.

Development Process :

The navigation bar should have a link to the homepage.

Estimated Time to Completion :

5 hours

Actual Time to Completion :

5 hours

Name : **20**

Book Attributes

Assignee :

Vishruthi Ramaswamy

Description :

As a user, I would like to receive specific information on books such as the Book Summary, Book's Genre, Book Image, Recommended Books in Genre, Publisher, Number of Pages, Rating, & Published Date.

Development Process :

Pull information from Database about Genre, Image, Publisher, Number of Pages, Rating, & Published Date.

Estimated Time to Completion :

6 hours

Actual Time to Completion :

6 hours

Name : **21**

Author Attributes

Assignee :

David Day

Description :

As a user, I would like to receive specific information on authors such as the Author Image, Author's writing Genres, Number of Books by Author, Recommended Books by Author, Publisher for Author, Recommended Authors in Genre.

Development Process :

Pull information from the database about genre, number of books by author, recommended books, publisher, recommended authors.

Estimated Time to Completion :

6 hours

Actual Time to Completion :

6 hours

Name : **22**

Genre Attributes

Assignee :

Matthew Jiang

Description :

As a user, I would like to receive specific information on genres such as the Genre Description, Number of Authors in Genre, Number of Books in Genre, Recommended Books in Genre, Recommended Authors in Genre.

Development Process :

Pull information from the database about the number of authors in genre, genre description, and books in genre.

Estimated Time to Completion :

6 hours

Actual Time to Completion :

6 hours

Name : **23**

Sorting

Assignee :

Siddhartha Shetkar

Description :

As a user, I would like to have the option of sorting search results by date & alphabetical order to easily organize my results.

Development Process :

When processing queries, return results with parameters specified by the user (sorting - alphabetical or date).

Estimated Time to Completion :

6 hours

Actual Time to Completion :

6 hours

Name : **24**

Recommended Book Information

Assignee :

Kumaran Arulmani

Description :

As a user, I would like to know the title & author of a recommended book in addition to the image displayed in the carousel of recommendations.

Development Process :

In the recommended books carousel, make sure to include the title & author of the book in the carousel.

Estimated Time to Completion :

6 hours

Actual Time to Completion :

6 hours

Name : **25**

Genre Scope

Assignee :

Jaino Vennatt

Description :

As a user, I would like to be able to view multiple genres (close to 30-50 genres) to be able to search for a variety of books and be introduced to new genres.

Development Process :

Search for specific authors in different genres and write scripts to add those to the database.

Estimated Time to Completion :

10 hours

Actual Time to Completion :

10 hours
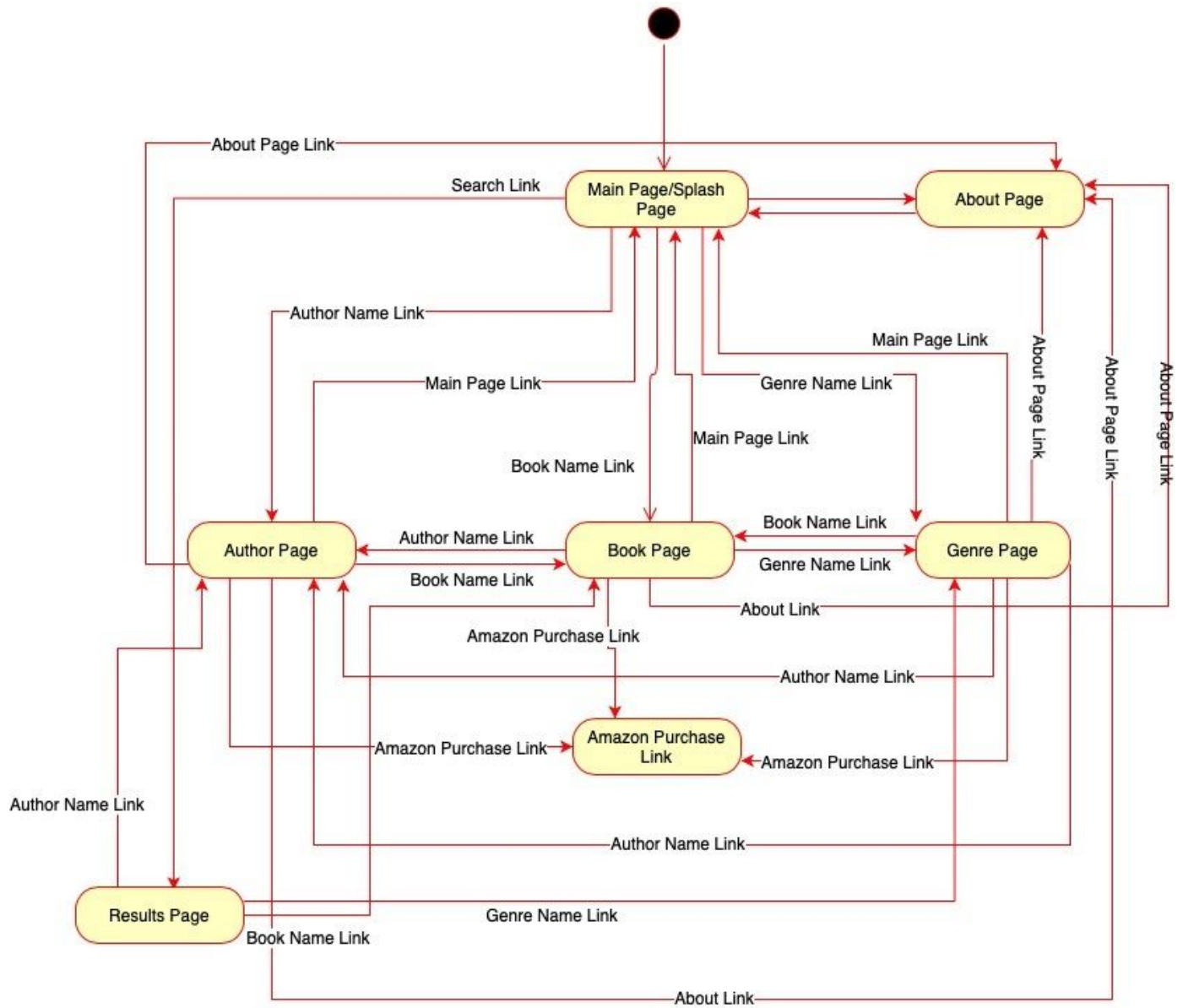
UML Diagram :
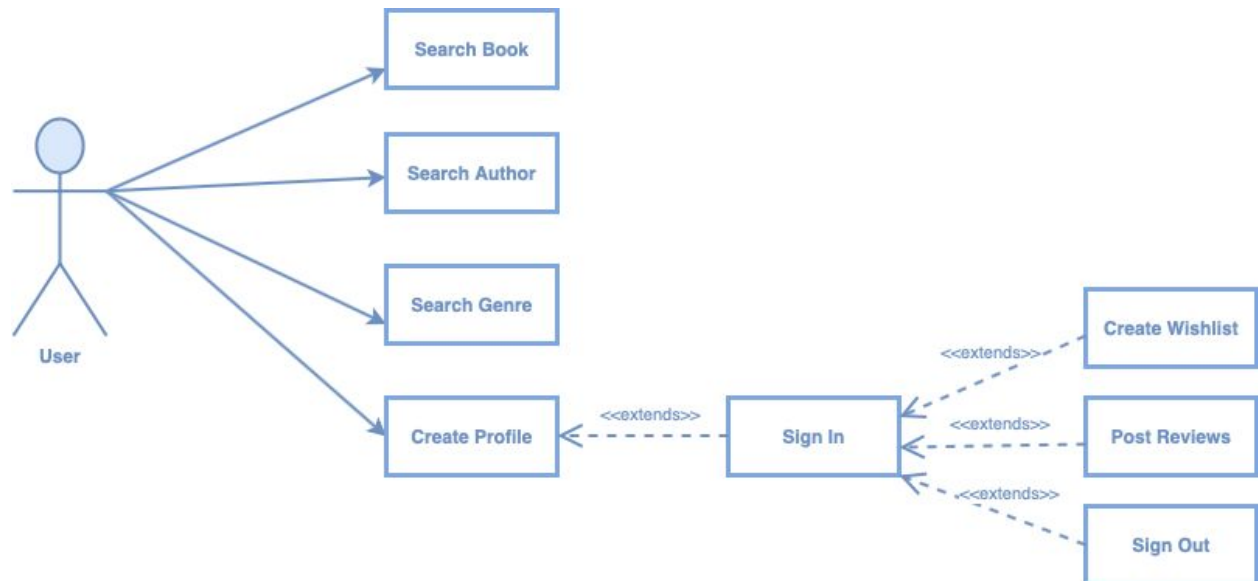
Design :

Our website starts on the splash page, where we plan on including the following attributes: a navigation bar (links to the home page and about page), and explore links (to different books, genres, and authors). We plan to have the navigation bar available on every page. Other pages we also have are book, genre, and author pages. Each of these pages will be dynamically generated from a base template and offer detailed information on each particular instance. Our base template for the author, genre, and book pages include author biography description, reviews, purchase links, genre description, popular books, popular authors, and book summaries. These pages are designed to be accessed through links on the homepage (editor's choice and explore) as well as through the search functionality. Our search functionality (which will be available through the Navigation Bar) will lead to a Results page - which will display the results of a query in a formatted order (pagination will be available on this page). Each result displayed will be a link to that particular model's page (a search for "J.K. Rowling" should return a link to the "J.K. Rowling" Page where the user may obtain more information about the specific model).

Implementation :

Phase I

We implemented static web pages for each of our models and had all overlapped information links to the corresponding webpages. For example, an author's name on a book web page would link to the web page for that author. We also queried the Github API for commit/issue/unittest information about each of our team members. Finally, we implemented a sidebar for easy access between the web pages of our models.

Phase II

We implemented dynamically generated websites (based on a generic template for each model). We also implemented query calls to the database to get information about a specific model instance. These query calls were paginated on the server side because it would slow down the website to send back all the data in one response. For example, to query all books would return almost 25000 books which severely impacted load time. We filled the MongoDB database with information about our models using our REST APIs through python scripts. We also implemented a lot of new intensive testing for our Front End, Back End, & Middleware.

Phase III

We implemented a wider range of genres available in our database. We also implemented more attributes for each of our models. We filled the MongoDB database with information about our models using our REST APIs through python scripts. We also implemented a lot of unit testing involving the GUI - to test all the new features implemented in this phase. In addition, we continued to refine our existing features to improve the user interface and make the site more navigable (grid like search results, carousels, etc.).

Testing:

Phase I

All of our testing for Phase I was completed by hand and user testing. All pages implemented thus far have been static (apart from the "About" page) so to test the feasibility of this page we viewed the pages on different screen sizes (laptop, tablet, phone) to make sure content was resized correctly (and dynamically). We also monitored the About page throughout the course of our working sessions to see the real time update of information. We hope to do more intensive testing in the upcoming phases once our pages are dynamically produced.

Phase II

With our pages being dynamically produced, we were able to implement more intensive testing in this Phase from the back end (database) to middle end (java/javascript that allows us to query the database) to front end (web pages/ui design). We tested both our front and middle end using a combination of Jest and Enzyme. Combining the libraries helped us test queries for the database

to see if we could pull stored data as well as simulate accessing web pages. We additionally used Postman to test the backend to make sure database entries were valid and exceptions were handled appropriately.

Phase III

During this phase we designed integration tests to ensure that our backend and frontend were functioning correctly. We designed these tests using the Mocha Framework, and WebDriver IO. We designed tests to test a wide range of inputs to see if it would break the website & if adequate error messages were shown. We also created tests to ensure that all our links/buttons in the front end are working correctly & leading to the corresponding pages.

Jest/Enzyme:
- Search Bar
- Re-rendering page correctly updates components after additional server call
- Improper URLs do not break the website
- Individual React components work as intended
- Server calls can give us dedicated information based on book/author/genre
- Server side pagination

WebDriverIO:
- Links on HomePage lead to the correct pages
- Search Bar is working correctly & thoroughly (can accept a wide range of inputs & behave appropriately)
- Search Results lead to correct links with the correct populated information on the Result Card as well as on the specific Model Page

Postman:
- Database Testing Phase 1
  - Put entry in database, check if entry is actually in the database
  - If there is no such entry in database, errors handled appropriately
  - Test for duplicate entries in database
- Backend API Testing Phase 2
  - Test every back-end API Call possible and check returned values
  - Double check that correct values are returned for random search
  - Double check that correct values are returned for specific search
  - Double check values for calls regarding books, authors, and genres

Back End API Test Example 1:

- Inputs:
  - Type - specifies model
  - Query - string to search
  - Page Number - specifies which page of results/ pagination
- Outputs:
  - JSON Object - array of all books pertaining to search as well as all fields regarded to each book
- Goals:
  - Check if our query returns appropriate book results for the sent query

Back End API Test Example 2:

GET   http://34.71.147.72:80/book?isbn=9781602062894   Send ▾   Save ▾

Params ●   Authorization ●   Headers (7)   Body   Pre-request Script   Tests   Settings   Cookies Code

Query Params

	KEY	VALUE	DESCRIPTION	••• Bulk Edit
☑	isbn	9781602062894		
	Key	Value	Description	

Body Cookies Headers (7) Test Results   Status: 200 OK Time: 182ms Size: 1.24 KB   Save Response ▾

Pretty   Raw   Preview   Visualize   JSON ▾

```
1   {
2       "_id": "5e816a419ff33650c88bef2a",
3       "volumeInfo": {
4           "publisher": "Cosimo, Inc.",
5           "description": "Classic of science (and mathematical) fiction -- charmingly illustrated by author -- describes the journeys of A. Square and his adventures in
                Spaceland (three dimensions), Lineland (one dimension) and Pointland (no dimensions). A. Square also entertains thoughts of visiting a land of four dimensions --
                a revolutionary idea for which he is banished from Spaceland.",
6           "publishedDate": "2007-06-01",
7           "imageLinks": {
8               "smallThumbnail": "http://books.google.com/books/content?id=4xiyEqfqh88C&printsec=frontcover&img=1&zoom=5&edge=curl&source=gbs_api",
9               "thumbnail": "http://books.google.com/books/content?id=4xiyEqfqh88C&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api"
10          },
11          "title": "Flatland",
12          "numberOfRatings": 10,
13          "pageCount": 108,
14          "industryIdentifiers": {
15              "identifier": "9781602062894",
16              "type": "ISBN_13"
17          },
18          "authors": "Edwin Abbott Abbott",
19          "genre": "scienceFiction",
20          "categories": [
21              "Fiction"
22          ],
23          "averageRating": 4
24      },
25      "saleInfo": {
26          "listPrice": {
27              "amount": 4.99
```

Bootcamp   Build   Browse

- Inputs:
  - ISBN - specifies unique ID of book being searched

- Outputs:
  - JSON Object - returns the book(singular) that matches the unique ISBN as well as all the values associated with this book
- Goals:
  - Check if our query returns appropriate book result for the sent query

Front End API Test Example 1:

```
JS Author.test.js
EE461L_IDB ▸ react ▸ src ▸ test ▸ JS Author.test.js ▸
  1 import React from "react";
  2 import { shallow, mount } from "enzyme";
  3 import Author from "../components/Author";
  4
  5 test("render Author component", () => {
  6   const componentDidMountSpy = spyOn(Author.prototype, "componentDidMount");
  7
  8   const wrapper = shallow(<Author />);
  9
 10   // ensure method was called
 11   expect(componentDidMountSpy).toHaveBeenCalledTimes(1);
 12
 13   //expect(wrapper.state('authorName')).not.toBeNull();   // currently null since authorName gets set by window.location.search
 14   expect(wrapper.state("authorPicture")).not.toBeNull();
 15   expect(wrapper.state("authorBio")).not.toBeNull();
 16   expect(wrapper.state("bookRecommendations")).not.toBeNull();
 17 });
 18
```

- Goals:
  - Check if Author component states are initialized properly.

Front End API Test Example 2:

```
JS Book.test.js     JS About.test.js     JS Description.test.js
EE461L_IDB ▸ react ▸ src ▸ test ▸ JS Description.test.js ▸
  1 import React from "react";
  2 import { shallow , mount } from "enzyme";
  3 import Description from "../components/Description";
  4
  5 test("render Description component", () => {
  6   const wrapper = shallow(<Description image="" description="test" />);
  7   const compInst = wrapper.instance();
  8
  9   expect(compInst.props.image).toEqual('');
 10   expect(compInst.props.description).toEqual('test');
 11 })
```

- Goals:
  - Check properties passed into the Description component are set correctly.

GUI Test Example 1:

```
1    /**
2     * Test that Books Card Media directs to right page on click.
3     */
4
5    const assert = require('assert')
6
7
8    describe('Booklopedia Homepage Books Card Media', () => {
9        it('should link to Books page on click', () => {
10           browser.url('http://localhost:3000')
11           const bookModelCard = browser.react$('CardMedia', {props: {title: 'Books'}})
12           bookModelCard.click()
13           const booksLink = browser.getUrl()
14           assert.equal(booksLink, 'http://localhost:3000/results?type=book&query=')
15       })
16   })
```

- Goals:
  - Check to see if clicking on links/cards on the HomePage leads to the corresponding Pages (About Us, Generic Books Page, Generic Authors Page, Generic Genres Page).

GUI Test Example 2:

```
/*
TEST 1 — NavBar Book Filter
Description:
Check to see if clicking the Navigation Bar's Book Filter without any input leads to
the generic Book Search Result Page.
*/
describe('Booklopedia Navigation Bar [BOOK]', () => {
    it('should be the generic book model page', () => {
        browser.url('http://localhost:3000')
        const bookButton = browser.react$('ListItem', {props: {id: 'books'}})
        bookButton.click()
        const newUrl = browser.getUrl()
        assert.strictEqual(newUrl, 'http://localhost:3000/results?type=book&query=')
    })
})
```

- Goals:
  - Check to see if clicking on Filtering Buttons (without any Input) takes you to the Generic Model Search Results Page.

GUI Test Example 3:

```
/*
TEST 3 — NavBar Book Filter w/ Special Character Input
Description:
Check to see if clicking the Navigation Bar's Book Filter with a special character user input leads to
the corresponding Book Search Result Page.
*/
describe('Booklopedia Navigation Bar [BOOK] — Special Character User Input', () => {
    it('should be the corresponding book results page', () => {
        browser.url('http://localhost:3000')
        const searchField = $('#search-bar')
        searchField.setValue('@#*')
        const bookButton = $('#books')
        bookButton.click()
        const firstSearchResult = browser.react$('Result').isExisting()
        assert.strictEqual(firstSearchResult, true)
    })
})
```

- Goals:
  - Check to see if clicking on Filtering Buttons along with specified input takes you to the tailored search results page.
  - Check to see if Search Functionality behaves appropriately for a wide range of inputs.

Models:

1. Books
   - Number of Instances: ~25000
   - Attributes: Author, Publisher, Genre, Cover, Year
   - Examples: The Road, Kite Runner, The Book Thief
2. Authors
   - Number of Instances: ~1000
   - Attributes: Name, Biography, Most Popular Publications
   - Examples: J.K. Rowling, Cormac McCarthy, Markus Zusak
3. Genre
   - Number of Instances: ~5
   - Attributes: Description, New Releases, Most Popular Publications, Most Popular Authors
   - Examples: Non-Fiction, Romance

Tools/Software/Frameworks:
We used the Google Cloud Platform to deploy the website. We used React (implemented with JavaScript) for our front end design and the Material-UI library to handle styling aspects of the website. MongoDB has been set up and filled with data using a combination of PyMongo and our REST APIs (Google Books, GoodReads, ISBN). We later pulled this data dynamically into our website by integrating NodeJS and Express with our frontend. The database will additionally hold user information (login, profile, wishlist, preferences) which we aim to complete in the next phase. For testing we used Jest, Enzyme, Postman, and WebDriverIO.

Reflection:
Phase I
Overall, our team got together with a great mindset and was able to hold a long productive working session that resulted in the majority completion of Phase 1. It was helpful to sit together and work in the same room, so many team members could pair program pages, ask questions freely, and bounce ideas off of each other. Our slack also helped us stay extremely organized throughout this process - with the ability to view git commits instantly from your phone, talk to teammates about pressing issues, create polls to vote on any team decisions. We were able to work independently after our joint working session, however productivity seemed to lag since members had to wait for the response of others through slack to make any design decisions/move forward with their assignments. For the upcoming phases, it would be beneficial for us to reduce the amount of independently worked time on the project and hold more joint working sessions to keep increased productivity.

Phase II
Overall, our team was able to pull together all the requirements specified for Phase II. We ran into some unprecedented issues (a World-Wide Pandemic) but our team was able to make the best of the situation. Through the use of Zoom, we hosted daily standup meetings, and longer working sessions (specific to our divided teams: BackEnd/MiddleEnd & FrontEnd). We had a great deal of work to be done on both teams and through the use of virtual meetings we were able to collaborate (screen sharing, chats). Our slack also helped us stay organized by giving us immediate updates on any issues, commits, and progress. Unfortunately with the situation at hand, we were not able to hold as many joint working sessions (virtually) as we would've liked, but had to each remotely complete our assigned tasks. As a result of this, our progress seemed to be slower than our previous Phase. In addition, our assigned tasks for this Phase were a great deal harder and complicated compared to the previous Phase. Switching the UI design (using React & Material UI) and querying the database were tasks that we were all not well versed in, so the learning curve was steeper. In conclusion, our team was able to meet the requirements for Phase II but lacked efficiency. We plan to improve this issue in future phases by planning a more strict timeline with clearly assigned tasks from the beginning that allows room for unprecedented issues and hiccups along the way.

Phase III

Overall, our team was much more organized and able to work more efficiently compared to Phase II. Working remotely still posed a couple of roadblocks for us, as we were unable to be there for each other all the time. However, we communicated as much as possible and were able to divide the tasks at hand pretty evenly & distributed the workload such that we were able to finish in a timely manner. This phase in addition to the couple of notes we received from our client group as well as our TAs, involved a lot of refinement of pre existing features. This being said, our dynamic for this phase was a bit more relaxed - as improving the features we had already spent quite a while developing was an easier task than to start from scratch. We did face some issues however when it came to the database, as we did not have enough space to include all the books we needed to expand our genre selection. We used the extra slip days this phase to further explore options on how we could solve this issue while still meeting the requirements of our project. In conclusion, our team was able to meet the requirements for Phase III, and improved a lot in terms of efficiency. We plan to further improve ourselves by communicating clearly, and dividing the work in a similar fashion (as we did throughout this Phase).