How can image recognition be utilized to allow students to gather information from books easier?

By: Yajat Sharma

Abstract

Students of all ages from around the world call upon books for information. Some need them for homework, some need them for school projects, and some need them for their own leisure. No matter the case, information needs to be extracted from the book. How my project aims to aid students in their use of books is through an app which displays information about the desired book to the user. Books related to the queried book are also shown. Information like the author, publication date, publisher, and a picture of the book will all be shown on the screen. Further, the user has the option to click a "Read Book" button which will take them to a website where the summary, rating, reviews, and a plethora more of information is stored.

Introduction

The vast majority of students in the world use textbooks. 75% of high school students in the United States reported that "overwhelming anxiety" was the main cause of negatively affecting their academic work. This anxiety and stress is usually due to the flood of work students receive from teachers every school day of the week. A lot of this work consists of using a textbook for information. For example, if a student has a book report as an assignment in an English Language Arts class, it might be hard for a student to choose a book that they are interested in. To properly determine whether a student will enjoy a book, they would most likely need to know who the author is, when the book was published, and most importantly, a short summary or preview of the book. An app which provides information about a book after just receiving a picture of the book title would be very useful. So, that is exactly what I created.

Overview

The app that I created allows the user to input an image of a book title and it gives the user the author, publisher, image, summary, and much more information about the book.

This app is a web application so it is easily accessible and is not limited by operating system or device. I made it with python as the main language, meaning that I used it for handling the server side and the functions of the app. I used html and css to create the website itself, not any of the processes. I used python because it is a very simple language and it has a very large user base. This made it very easy to debug. For example, if there was any issue with the code or something needed to be implemented, I could very easily ask a question on the StackOverflow website and it would be answered by one or more of the millions of other python users. Python also has a library called Flask, which is another reason why I chose to use it. Flask is used to create web-based applications with python. With other languages or other websites, the programmer would have to implement a separate server system that is not directly incorporated into their code. Having Flask allowed my workflow to be much smoother and much more simple to work with. There is also another way to use Python to make web applications. A package called Django allows for more functionality. Despite the wider range of functions, I decided to go with Flask because it is much simpler to work with and I had already seen a few tutorials about how to set up Flask. Django also had many features that I simply had no use for so it was not in my best interest to use it.

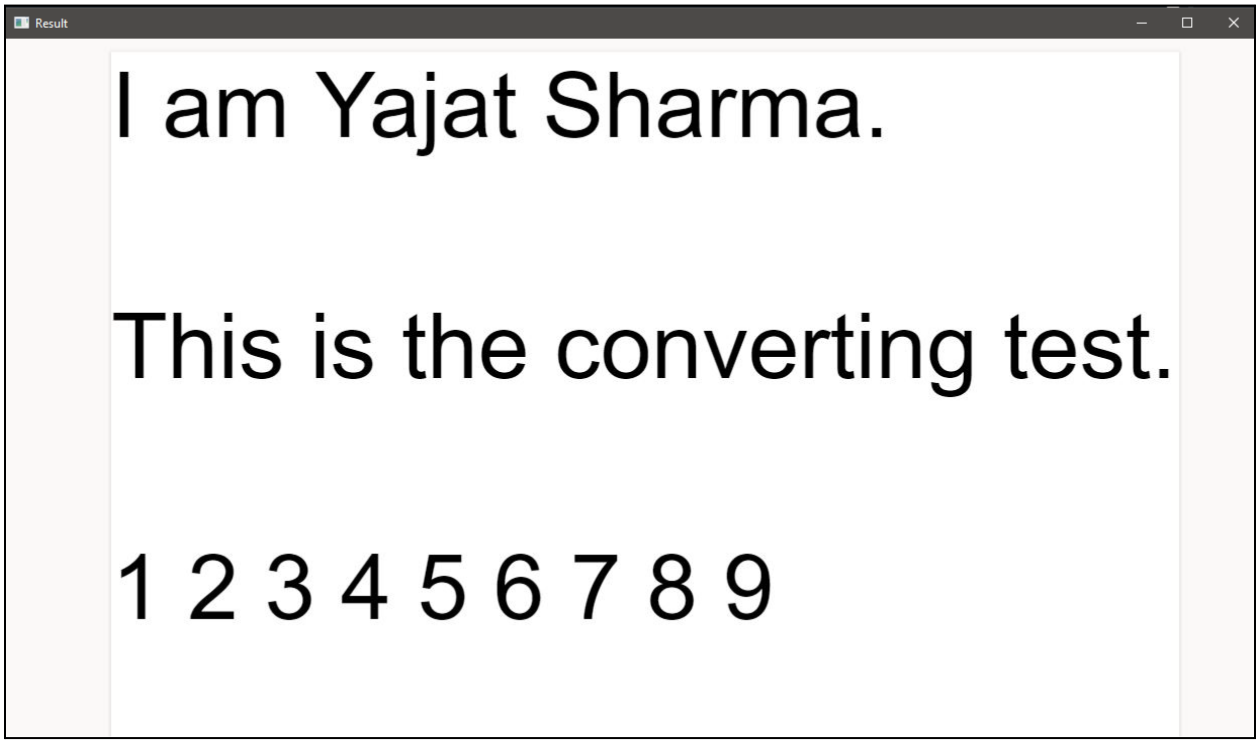
My main choice of IDE was Jupyter, in the form of a Jupyter notebook, Spyder, and PyCharm. I used a Jupyter notebook to easily display certain aspects of my project when I needed to. I then moved on to Spyder once I got to the Flask operations because it is easier to manage the server from a proper IDE. I then switched to PyCharm for ease of use and because Spyder was older-fashioned and the font was not as readable as PyCharm's.

Aside from the website side of operations, there is one more crucial part of the application. That is the text detector. Since the app takes in a picture instead of plain text, the words in that image need to be converted into text. To do this, I used Tesseract, which is an optical character recognition engine developed by Hewlett-Packard and Google. Tesseract OCR is 70% to 80% accurate in general but if the letters are very clear, the accuracy increases dramatically.

Process

The first step in creating this app was setting up a Flask system and a local server. For this I used a few tutorials on Youtube to learn how to properly use the Flask module. I also turned to multiple Medium articles in order to gain more knowledge and experience with python based web applications. Some of these articles are Easily Build Your First Web App With Python's Flask by Jonathan Hsu, A Complete Guide to Web Development in Python by The Educative Team, and others. Another valuable resource I utilized was stack overflow. While I was implementing certain parts of the app, I came across issues. Most of them were minor issues which other people also had while implementing other things in their own projects. So, I was able to ask questions and check existing questions on stack overflow in order to fix many of my bugs and implement new features.

The second step was integrating Tesseract into my website. At this point, I had a Flask web page working properly so I decided to work separately from the website side of operations so I can first understand how Tesseract even works so I can recombine the two processes later. Now, while working with Tesseract, I was unsure about how to even use it so I, once again, turned to Medium as my source of information. I used articles like Tesseract OCR for Text Localisation and Detection by Sharon Lim, How to Extract Text from Images with Python by Costas Andreou, and others. The first step in working with Tesseract was getting it to perform on some text, no matter how simple it is. First, I had to download Tesseract from GitHub. This is because the Tesseract is a whole engine and I cannot incorporate a whole engine into just one or a few files. So, I had to download the engine from the internet, then I had to reference it in my code so that it can be used from somewhere else in the computer. After this, came the testing. First, I ran it on just a still image. This was a success and the text was outputted to me in the console. The next steps, mainly for practice, were to make the text detection more user friendly. On every video that I had watched about AI that I had watched in the past, an example of text detection was shown. In every example, the text had boxes around each word and the detected words were written above. This is exactly what I wanted to do and is where my inspiration for this part came from. So, the next step was putting boxes around each of the words to make sure that the correct things were being detected. At first I had no idea how to do this but I researched extensively on GeeksForGeeks, YouTube, and Medium in order to figure out how to do this. Eventually, I came across a very informative YouTube video explaining a good way of doing what I needed to do. Without the boxes:

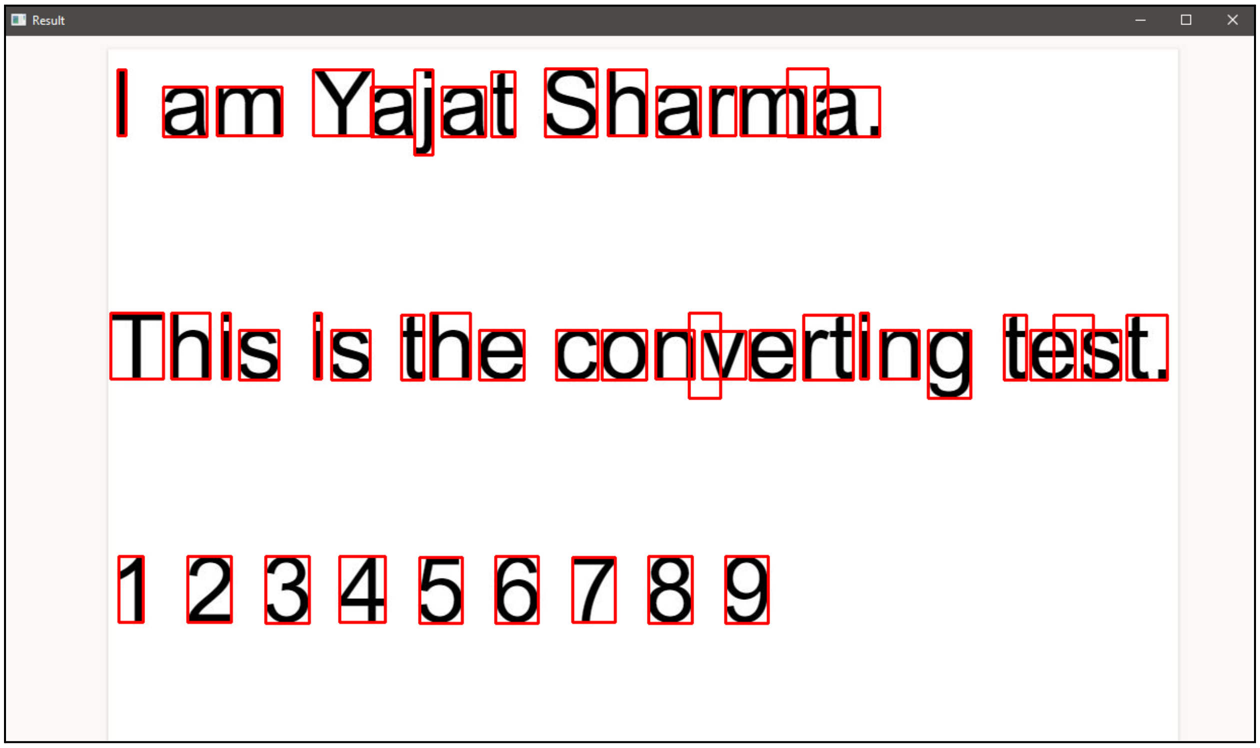


I am Yajat Sharma.

This is the converting test.

1 2 3 4 5 6 7 8 9

With the boxes:

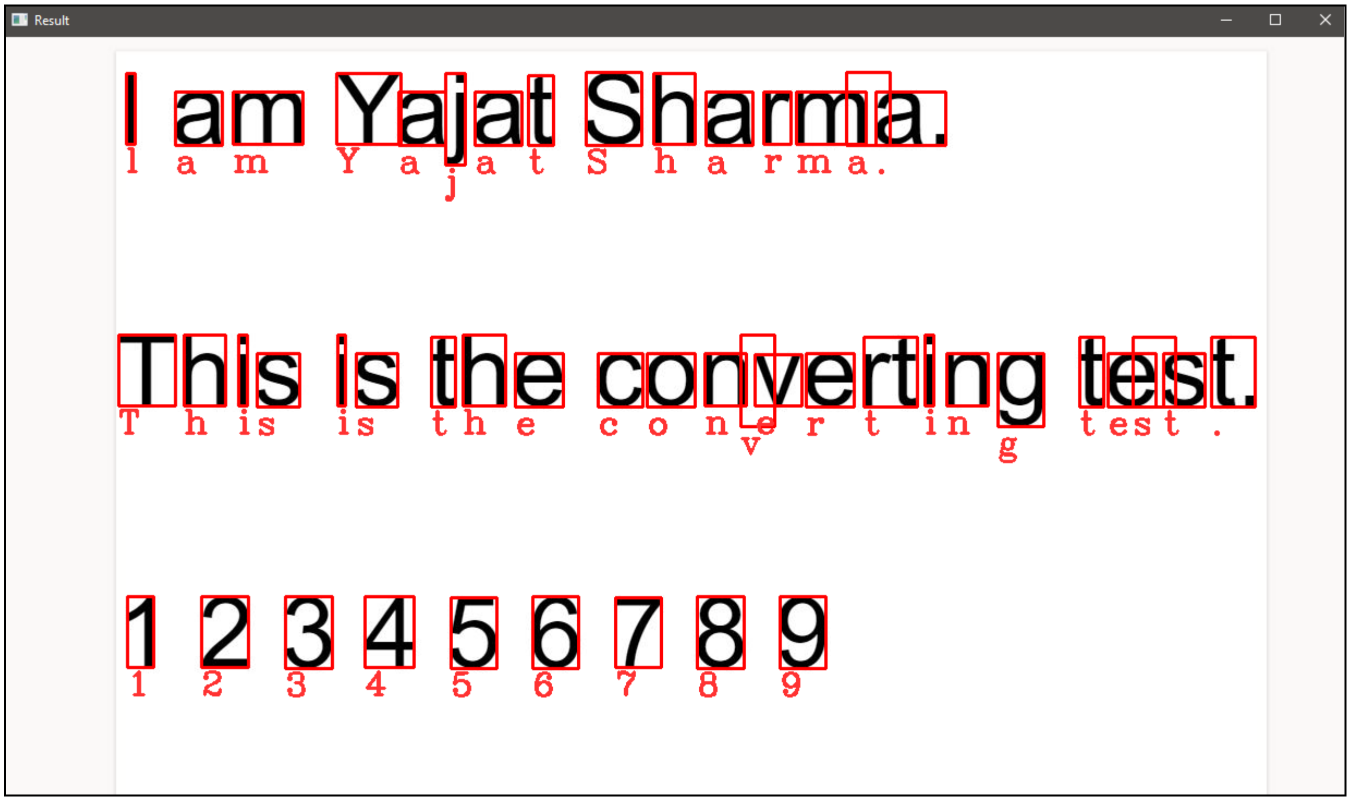


I am Yajat Sharma.

This is the converting test.

1 2 3 4 5 6 7 8 9

After this, I had to place the text there too. I did this in a similar fashion as the boxes. With text:



I am Yajat Sharma.

l a m Y a j a t S h a r m a .

This is the converting test.

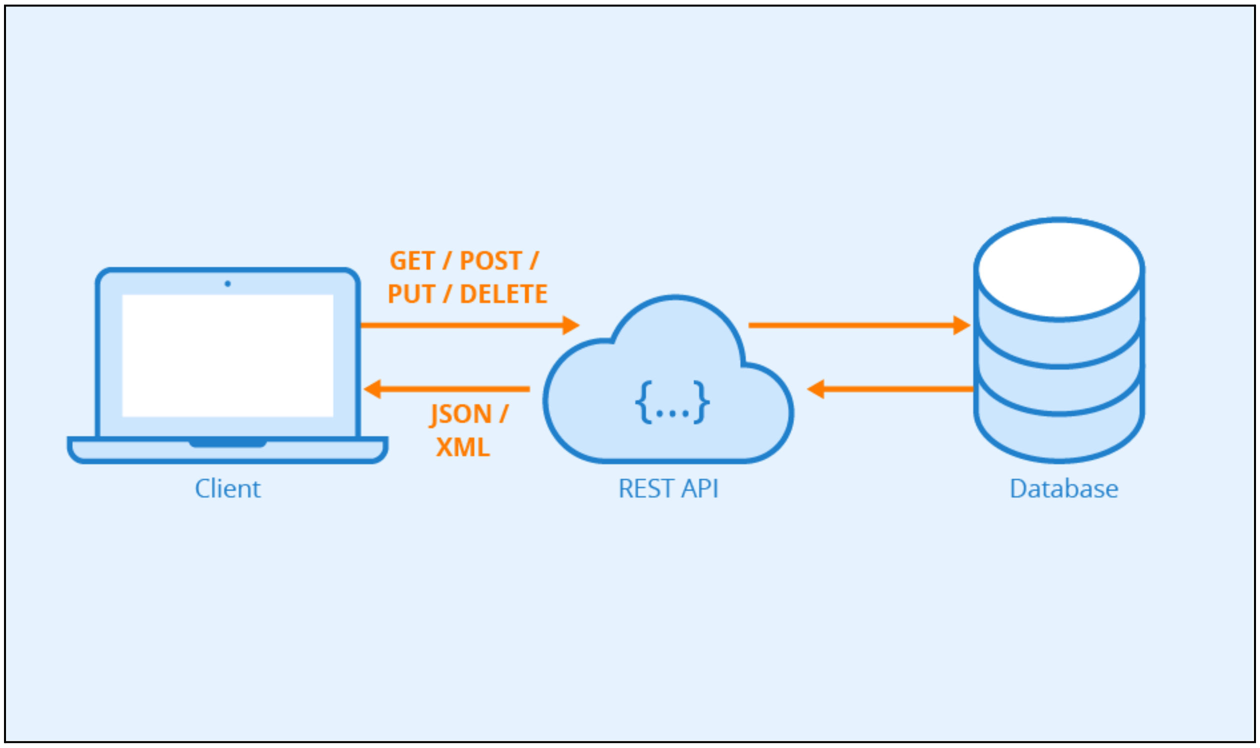
T h is is t h e c o n v e r t in g t es t .

1 2 3 4 5 6 7 8 9

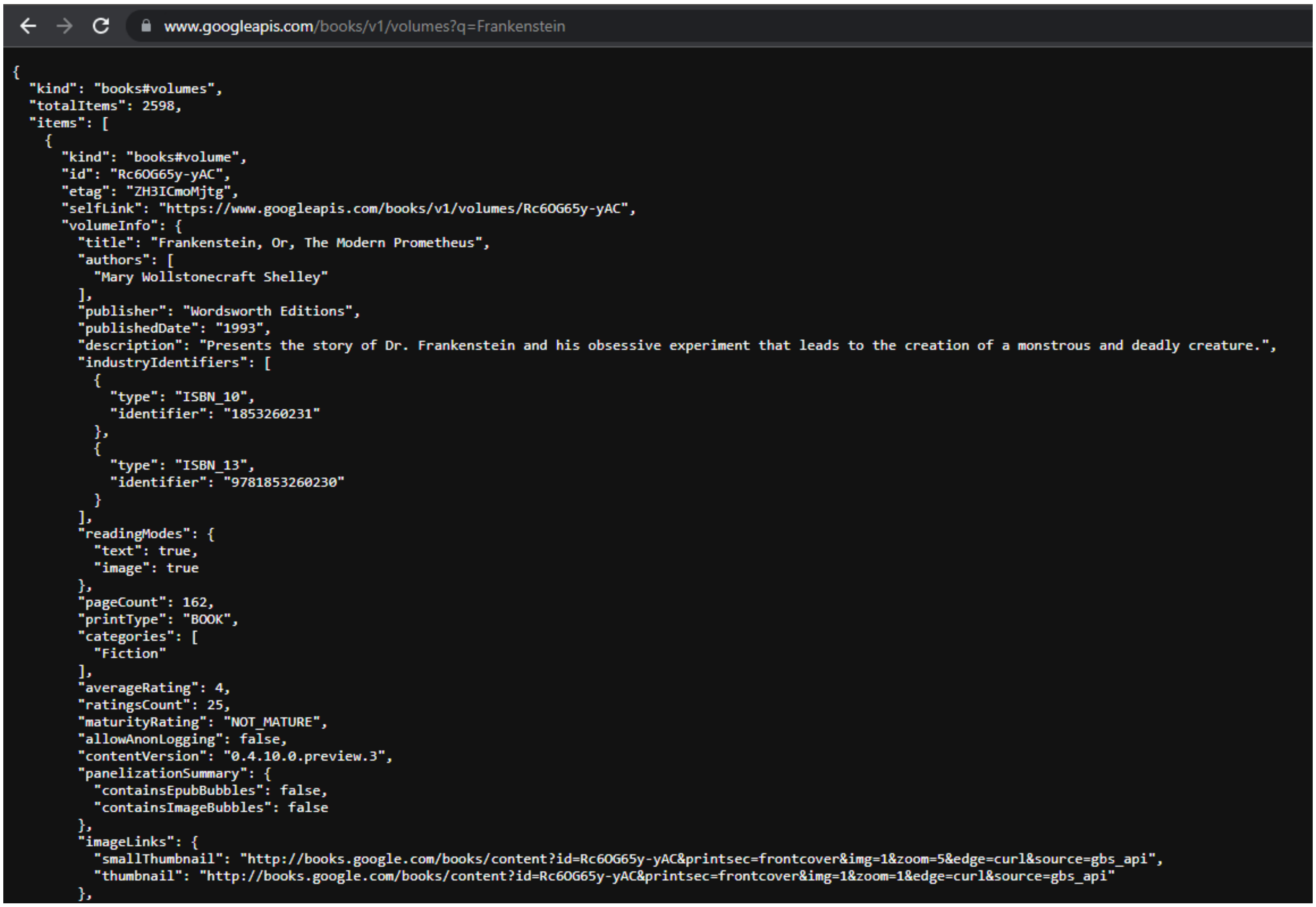
1 2 3 4 5 6 7 8 9

I tested this with many different books and pieces of text and they all worked very well, more so if the text was clean and unobstructed. Once testing of Tesseract was completed, I had to allow the user to actually use it with the website. Originally, I was aiming to implement a camera system with text detection and I was partially successful. The boxes were appearing around the text, but only for a very short time. It would then disappear and the camera would freeze. I did immense research on how to solve this problem. Despite this, I was not able to solve it, so I turned to the next best thing, file inputting. From Medium, I learned how file inputting systems can be implemented in python so I used that. The file that the user inputted then became the file that Tesseract was used on. In this stage I was having quite a few issues because I was not able to get the input file button to work properly. The file was not being sent to the code, it was static. I eventually did fix it and Tesseract performed on the image perfectly and returned the text in the console. Now I came across an issue, how will I even be able to use this text?

The third step in the process was actually using the text. What I wanted to do was use the text and search it up online in some way. I knew that I had to use a web API for this. API stands for "application programming interface". It is a way for two or more computers to communicate with each other through the internet. In this case, it is my computer, and whatever search engine will be used.



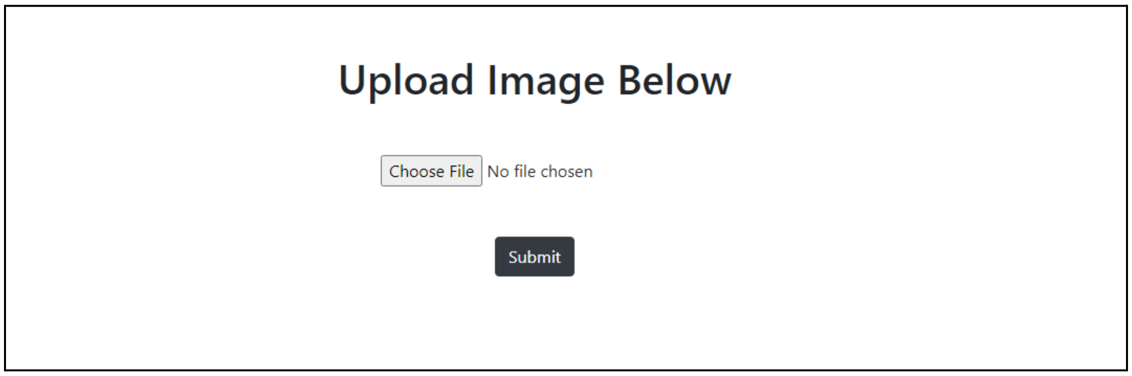
The communication happening is my computer is sending the text to be searched, and the API searches it up, and returns the results of the query to me. This is incredibly useful because the programmer does not have to write a whole new program just for one part of their original program, they can simply use something that someone has already made. There are many APIs on the internet, but I needed to find an API that would specifically search for books. For this, I looked at a few different APIs. One of them was the GoodReads API. It seemed very promising as it provided many features and methods of searching but unfortunately, the service was recently discontinued. Another book related API is the New York Times books API. I did not use this API because it did not provide enough functionality for my project. Its searches could only be of New York Times best sellers, and not all books fall under that category, so I had to move on. I then came across the Google Books API. This is the API that I ended up using. I ended up using this API because it had the exact functionality that I needed. I wanted to search for books based on text, Google Books API had exactly that. The format for the response of the query was in a JSON format (standing for JavaScript Object Notation), which looks like the following,



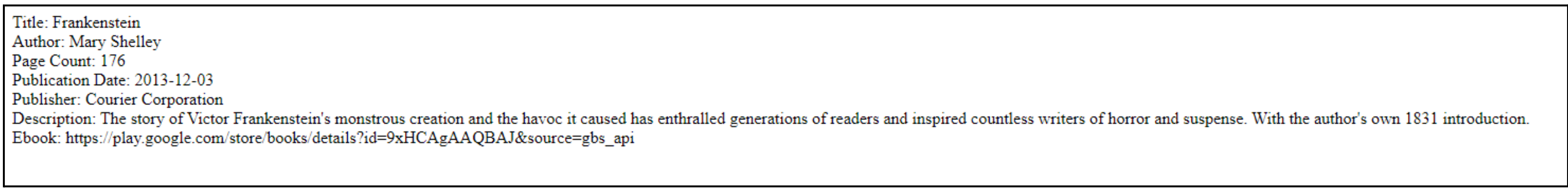
www.googleapis.com/books/v1/volumes?q=Frankenstein

```
{
  "kind": "books#volumes",
  "totalItems": 2598,
  "items": [
    {
      "kind": "books#volume",
      "id": "Rc6OG65y-yAC",
      "etag": "ZH3ICmoMjtg",
      "selfLink": "https://www.googleapis.com/books/v1/volumes/Rc6OG65y-yAC",
      "volumeInfo": {
        "title": "Frankenstein, Or, The Modern Prometheus",
        "authors": [
          "Mary Wollstonecraft Shelley"
        ],
        "publisher": "Wordsworth Editions",
        "publishedDate": "1993",
        "description": "Presents the story of Dr. Frankenstein and his obsessive experiment that leads to the creation of a monstrous and deadly creature.",
        "industryIdentifiers": [
          {
            "type": "ISBN_10",
            "identifier": "1853260231"
          },
          {
            "type": "ISBN_13",
            "identifier": "9781853260230"
          }
        ],
        "readingModes": {
          "text": true,
          "image": true
        },
        "pageCount": 162,
        "printType": "BOOK",
        "categories": [
          "Fiction"
        ],
        "averageRating": 4,
        "ratingsCount": 25,
        "maturityRating": "NOT_MATURE",
        "allowAnonLogging": false,
        "contentVersion": "0.4.10.0.preview.3",
        "panelizationSummary": {
          "containsEpubBubbles": false,
          "containsImageBubbles": false
        },
        "imageLinks": {
          "smallThumbnail": "http://books.google.com/books/content?id=Rc6OG65y-yAC&printsec=frontcover&img=1&zoom=5&edge=curl&source=gbs_api",
          "thumbnail": "http://books.google.com/books/content?id=Rc6OG65y-yAC&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api"
        },
```

making it very easy to navigate through the response. It has the author's name, the publishers, the publication date, and multiple pictures of the book itself. With all of these, I could make the website even more user friendly. Once I had figured out how to query the text and get back a result, I had to figure out how to get it onto the website. With Flask, since it is in Python, the print statement simply displays things to the console so that was not the way to put the text onto the website. I asked how to do this on Stack Overflow and received a response fairly quickly. What I had to do was just return everything from my function that I wanted to display. I was reluctant to do this because of how barren the website would look, but I forced myself to do it because I had no other option. And yes, I was right, the website was very unappealing. At first, I did not use another HTML file, so my website looked completely bare. Here is what it looked like:



After search:



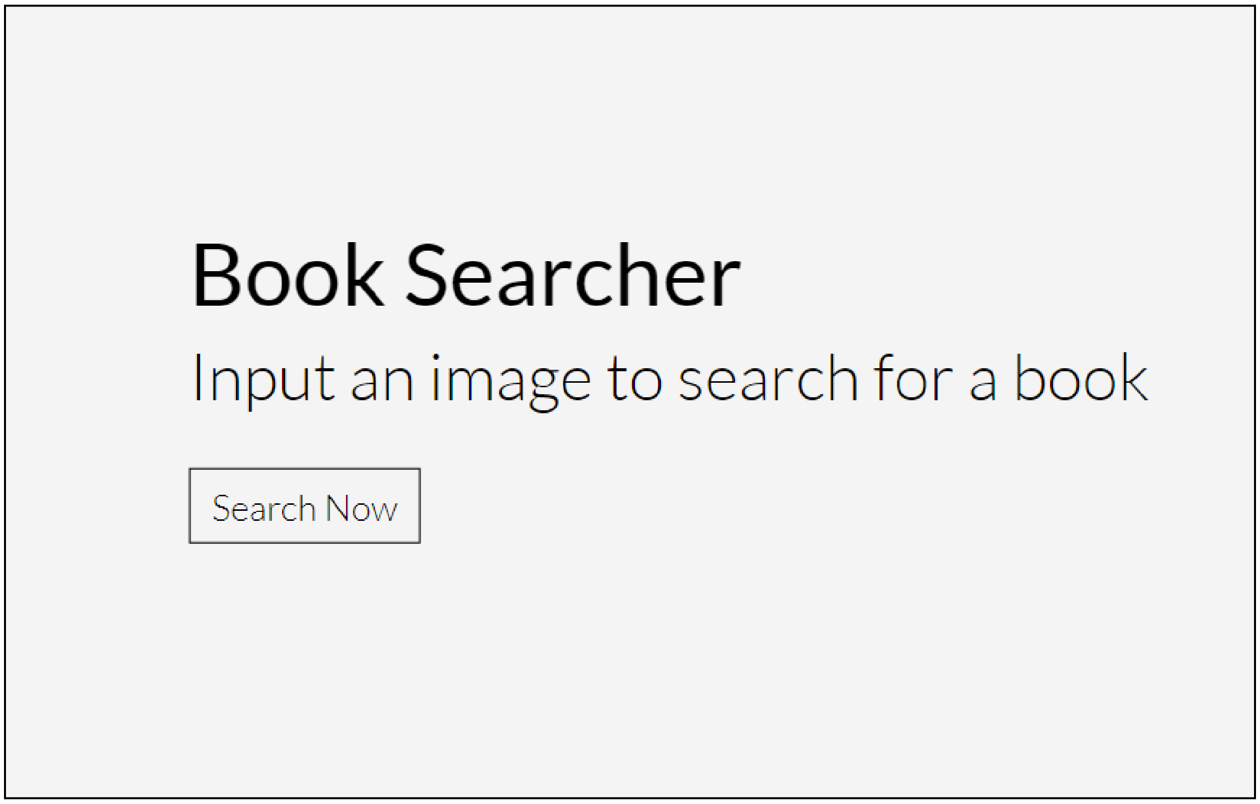
```
Title: Frankenstein
Author: Mary Shelley
Page Count: 176
Publication Date: 2013-12-03
Publisher: Courier Corporation
Description: The story of Victor Frankenstein's monstrous creation and the havoc it caused has enthralled generations of readers and inspired countless writers of horror and suspense. With the author's own 1831 introduction.
Ebook: https://play.google.com/store/books/details?id=9xHCAgAAQBAJ&source=gbs_api
```

I knew I needed to change something, but I had to implement one more feature first.

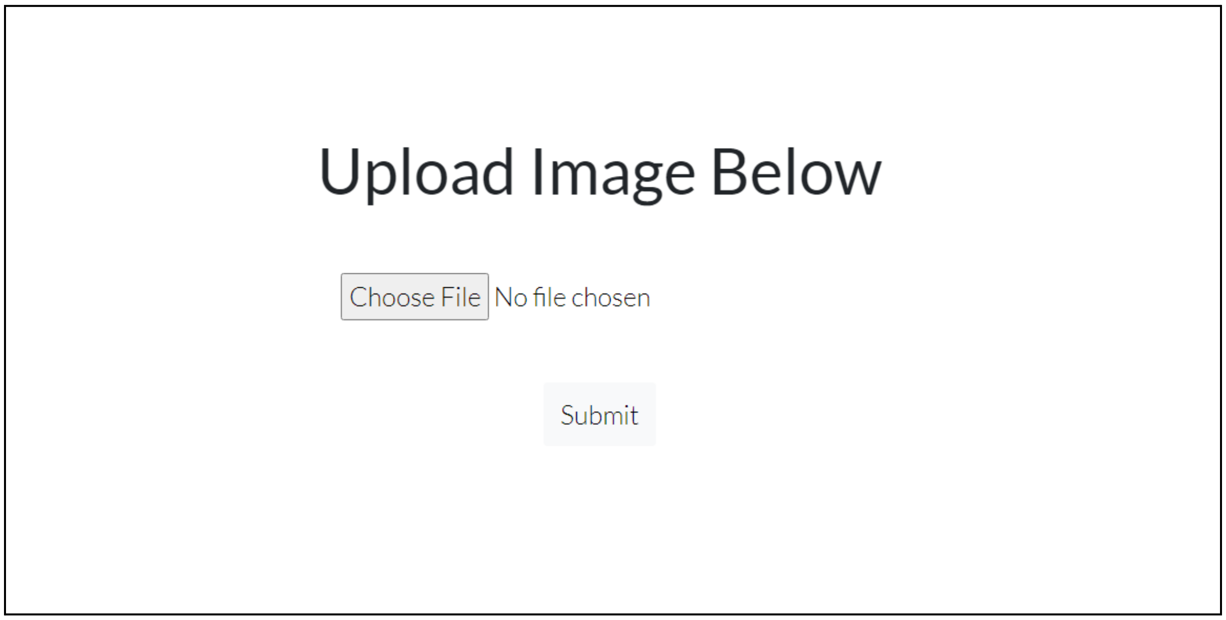
The fourth step in creating this website was adding a text search area. I decided to add a text search field because it would aid the user even further. For this, I had to use an HTML file. To link the HTML file with my python file, I had to return it as a render template, which I had never done before. I first started with a simple search box which the user can enter text into. Then I created a “search” button which would, essentially, submit the text. The hard part was getting the text onto the screen again. When I did this previously, I used Python which I was more comfortable with. Also, previously, I made the website switch to another one when I wanted the output to be displayed. I realized that if I could put the returned information on the same web page, it would be much more user friendly. So, that is what I set out to do. I had to do some JavaScript to tackle this task. It helped me immensely because it allowed my HTML to be non-static, which was an incredible improvement which increased the appeal and usability of the website. On a side note, one thing that also increases user productivity is that the query can essentially be anything the user wants. It can be the book name, author name, publisher name, or other things. Another idea which I had was to display all results of the query. For most queries, like Google for example, return many different things when the user queries just one thing. This is similar to what I wanted to do. Before, I had information of just one book being returned. Since the returned information was in a JSON format, I looked at the file and realized that what I wanted was possible because it had the information of multiple different books. This would expand the functionality of the website because not only does the user have information about the book that they wanted, but they also have information about a related book. Once I had multiple books returned, I needed to decide what information I wanted to display on the side panel. I decided to put the author name, publisher, and a picture of the book. In addition to this information, I added a “Read Book” button. What it does is take the ISBN (International Standard Book Number) of the book from the JSON file, adds that number to the end of a website link, and takes the user to that website. This website is special because I am able to search for a book by just adding the ISBN to the end of the link. The website is called “openlibrary.org”. It provides the rating, number of ratings, reviews, an image, pages, number of editions, related books, and so much more. This further improved user productivity because they have access to a vast range of information.

The shorter, but still important fifth step in the process was adding the same functionality as the text search to the file-based search. Once I created the text search with all of its information and non-static functionality, I realized that it would be a much better interface than the static one that I had previously. All I had to do was create a new HTML file, copy the information from the original HTML file, and modify it slightly to work properly with the file-search implementation. To link the HTML file with the Python file, I had to return it as a render template again, but I had to add something. I had to pass the detected text to the “render_template” function. This is because my original HTML file was designed to work with a search bar, where the user inputs information which is processed through the HTML file directly. The problem was that I had the detected text in my python file, but I needed to use it in an HTML file. I thought long and hard about how to solve this. I eventually asked on Stack Overflow and saw that the solution was as simple as just a few words. All I had to do was say that the variable = the detected text. This was not the end to my problems though. When I tried to run my program, it still did not work properly. This is because to use a Python variable from a different file in HTML, a special notation is needed where the variable is written in double curly braces. Eventually, I did implement the improved interface but I wanted the result of the query to show up on a separate web page only for the file-based section because it is the main part of my website.

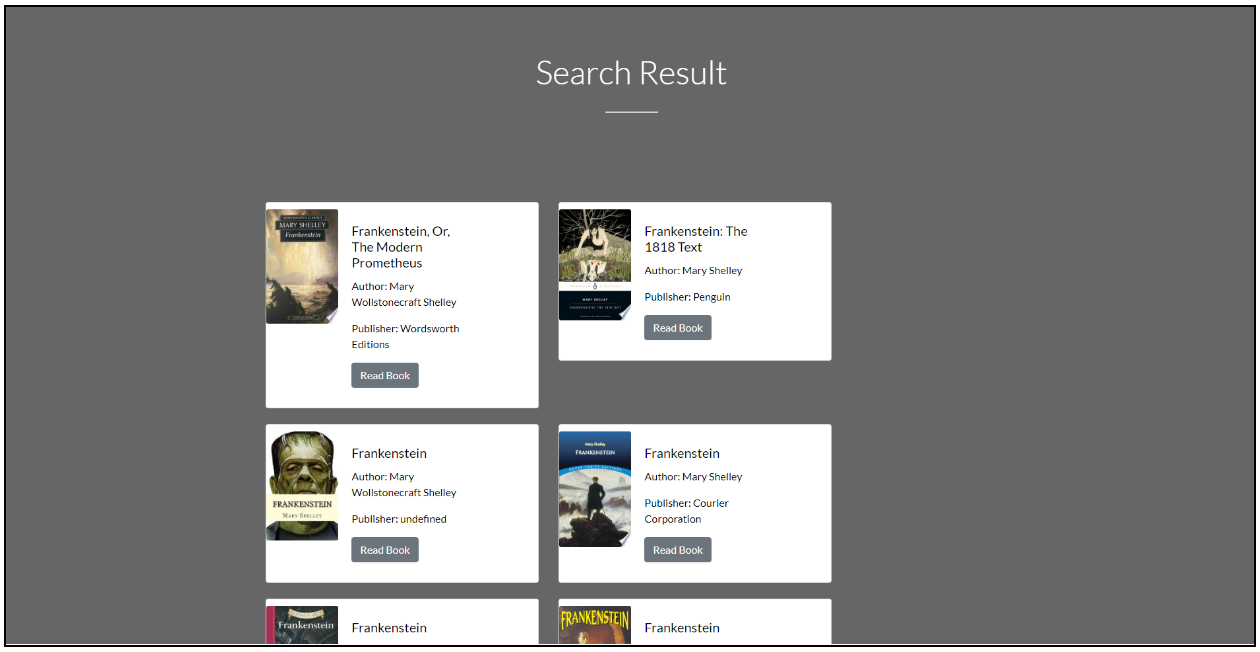
The immensely important sixth step to the process was improving my HTML and actually creating CSS for my web page. Before, my web page was very dry and boring, the JavaScript functionality made up for this slightly, but was just not enough. I needed my web page to look more professional. So, I wrote my HTML properly and created styling in CSS for everything on the web page. I made a first section to the website which has the title and a button which scrolls down to the next section. I changed the font, made it larger, and made the button look like a proper one. It looks like:



The next section is the file-based search section. Once a file is inputted and the "Submit" button is pressed , a new web page is loaded which displays all of the books found and their information. Before search:



After search:



Search Result

Frankenstein, Or, The Modern Prometheus

Author: Mary Wollstonecraft Shelley

Publisher: Wordsworth Editions

Read Book

Frankenstein: The 1818 Text

Author: Mary Shelley

Publisher: Penguin

Read Book

Frankenstein

Author: Mary Wollstonecraft Shelley

Publisher: undefined

Read Book

Frankenstein

Author: Mary Shelley

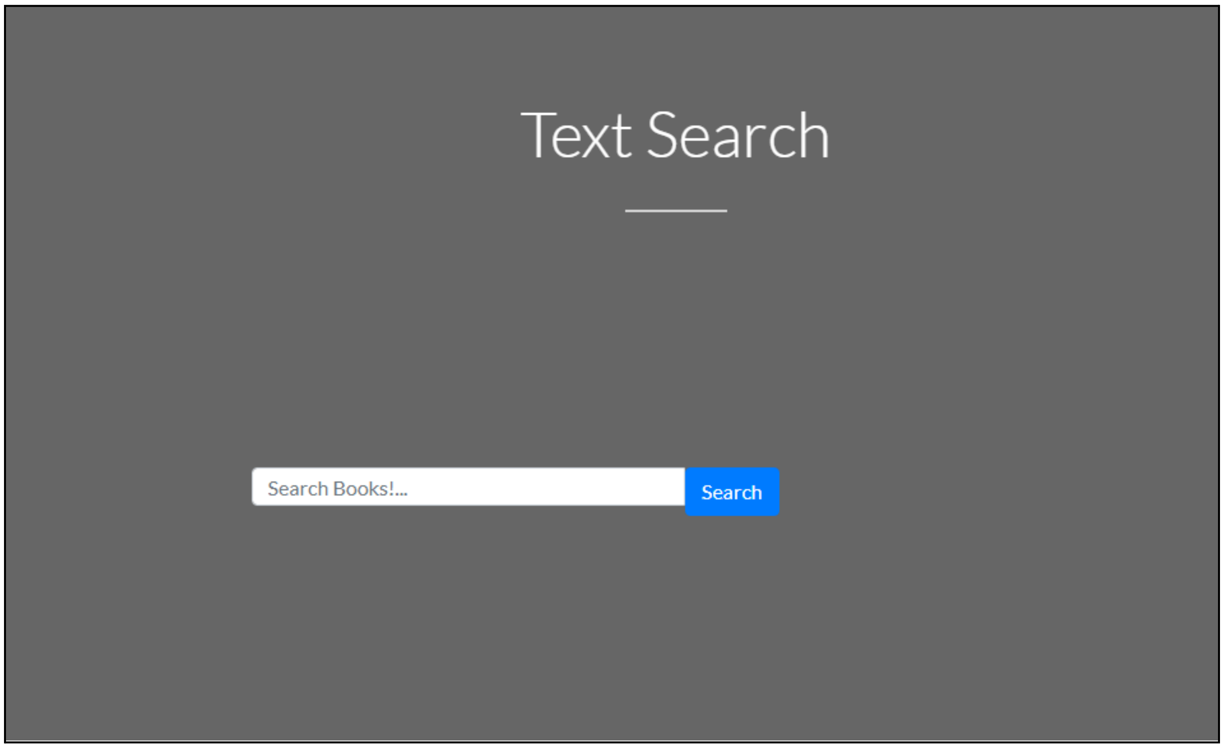
Publisher: Courier Corporation

Read Book

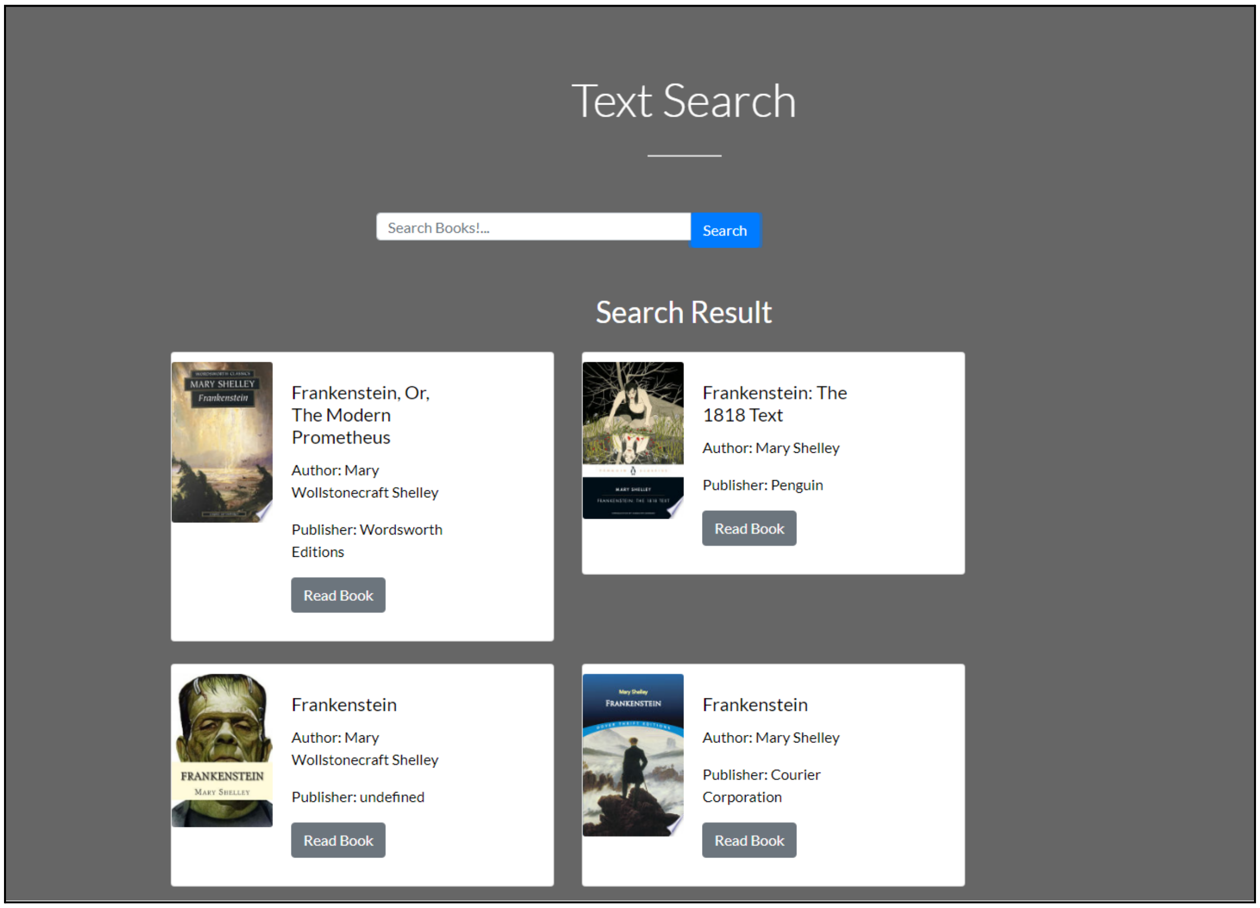
Frankenstein

Frankenstein

The third section on the main web page is the text-based search. When the user clicks "Search", the web page automatically scrolls down and the section increases in size to accommodate for the search results. Before search:



After search:


Text Search

Search Books!...

Search Result

Frankenstein, Or, The Modern Prometheus

Author: Mary Wollstonecraft Shelley

Publisher: Wordsworth Editions

Frankenstein: The 1818 Text

Author: Mary Shelley

Publisher: Penguin

Frankenstein

Author: Mary Wollstonecraft Shelley

Publisher: undefined

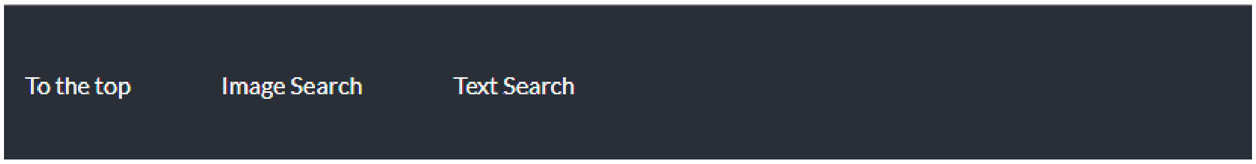
Frankenstein

Author: Mary Shelley

Publisher: Courier Corporation

I was not able to center the search bar properly because I needed to accommodate for the search results.

The final "mini" section is the footer. All it has is a button to go back to the top, a button to go to the file-based search, and a button to go to the text-based search. The footer looks like:



Debugging

Debugging is a crucial part of any program. It just means, finding any issue with the program and fixing it. I did indeed have my fair share of issues with my code.

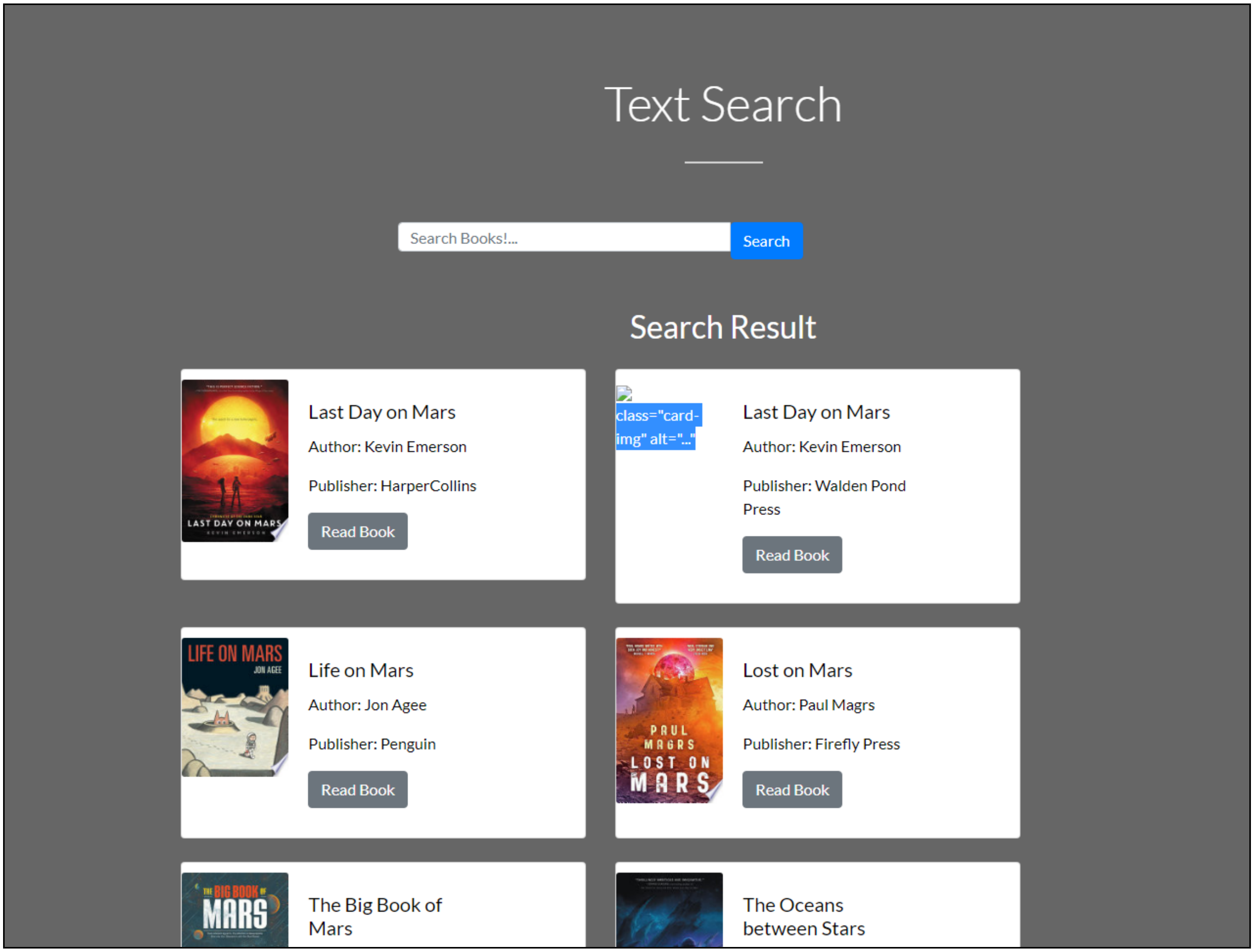
My first issue was in the first step of the process, the creation of the Flask system. For some reason Flask was simply not working on PyCharm. Even after importing the library and working through a tutorial on YouTube, I was still getting an error that said that Flask was not an existing library on my computer. I decided to switch to Spyder for the time being because it was connected with Anaconda which had an immense amount of libraries which came with it. Flask is a very popular library so there was no doubt that Anaconda would have it. I used Spyder and my Flask web page ran properly.

My second issue was in the second step of the process, a problem with Tesseract. This issue occurred while I was testing out Tesseract in a different file. When using Tesseract in Python, the following statement must be used:

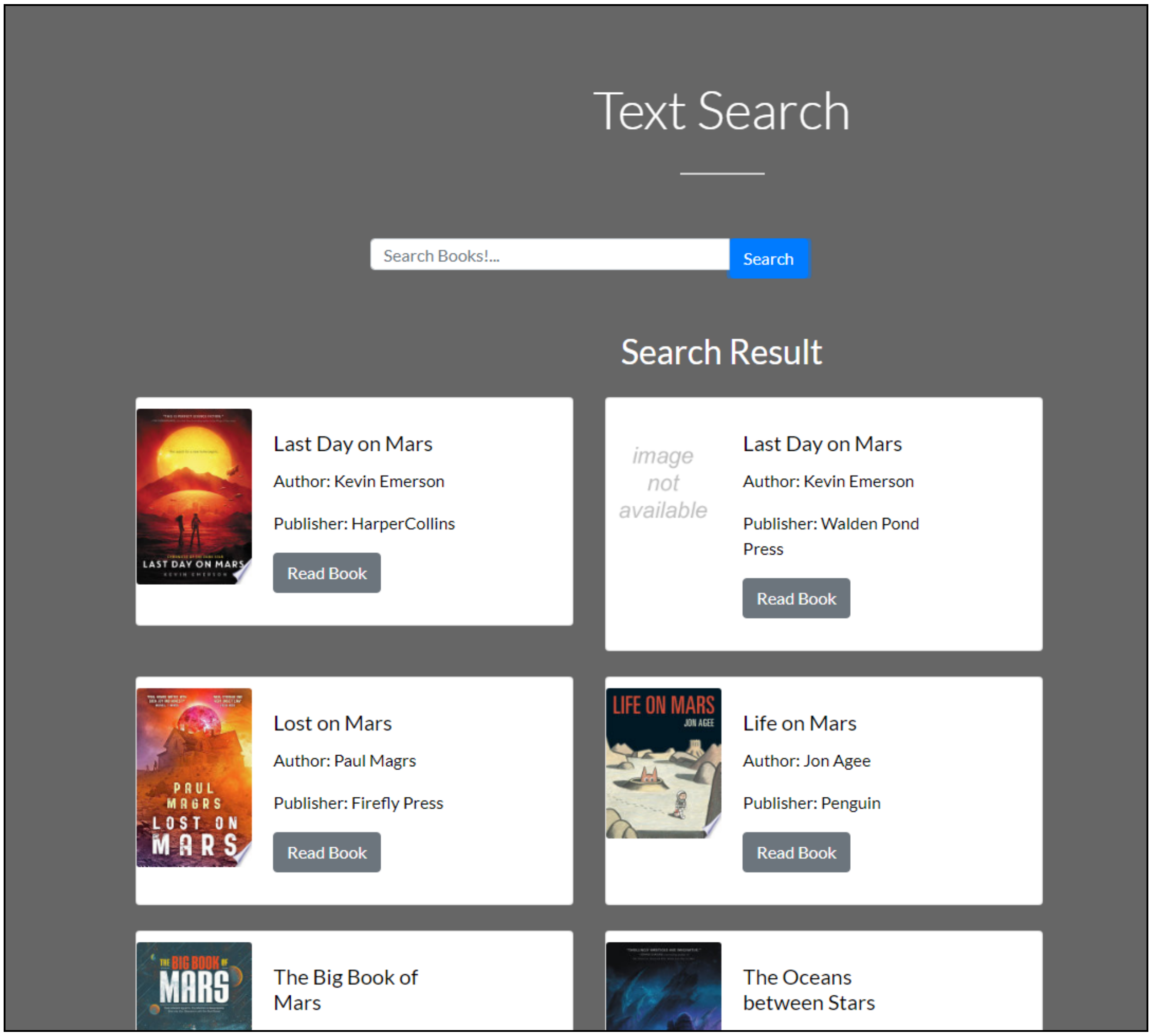
```
pytesseract.pytesseract.tesseract_cmd = 'C:\\Camera_Flask_App-main\\Tesseract\\tesseract.exe'
```

This statement is used so that the program can access the downloaded version of Tesseract on the computer. When I tried running a detection program with this, I got errors saying things like “no such file exists at this location”. I checked multiple times that Tesseract was at that location. I then decided that something else must be wrong so I deleted then redownloaded Tesseract at the same location and the program started to work. I assume something went wrong during the install process and maybe a few files were not downloaded.

My third issue was not a program-breaking issue, but it was annoying nonetheless. The problem was in the fourth step, when I was creating the text search section. The problem was that the Google Books API JSON file was not completely accurate. For each query, there are multiple books given, but some of the books did not have an images section. This caused a whole part of my HTML and JavaScript code to stop working. The problem was that since there was no images section, that area would be blank and something very weird would pop up. Upon querying “Last Day on Mars”, the following shows up:



Here, a default image shows up instead of the real one and white text which is part of my HTML shows up. I was very confused as to why this was happening. I thoroughly sifted through my code and found that the program was not properly handling a situation where there is no images section. I had a default image that was supposed to be shown in the case that there is no image. That was not being properly used. So, I added an “if” statement saying that if the image shown is the default image that I specified, show a different image. The addition of this statement produced the following result:


Text Search

Search Result

Last Day on Mars
Author: Kevin Emerson
Publisher: HarperCollins

Last Day on Mars
Author: Kevin Emerson
Publisher: Walden Pond Press

Lost on Mars
Author: Paul Magrs
Publisher: Firefly Press

Life on Mars
Author: Jon Agee
Publisher: Penguin

The Big Book of Mars

The Oceans between Stars

Here, the shown image is a default image saying “image not available”. There is also no white text under the image, so I know that I have fixed this bug.

Conclusion

From grades k-12, 70% to 85% of students reported using books for school. 75% of high school students reported that “overwhelming anxiety” as a result of their piles of school work is the main reason for their decreasing academic performance. These two statistics could definitely be linked. Stress correlated to the usage of books in school. An assignment which requires a book could be a book report where a student has to read a book of their choice and create a report about it. A student might also be having a hard time picking out a book to read, but they do know what kinds of books they enjoy. I created this website to try to alleviate problems like these. My website provides all basic information about a book such as the summary, page count, reviews, ratings, and where to access the book. All of these would greatly improve the productivity of a student writing a book report. The student can also search for a book that they enjoy and similar books will also be shown. This would help a student choose what book they want. I really enjoyed developing this website as I learned many different web-related methods such as Flask, HTML, CSS, JavaScript, and even a text detector, Tesseract. I hope to be able to create more projects in the future which can be used to better society as a whole.

Resources Used (Links)

Before production:
https://medium.com/@PeterBruce/matching-algorithms-d3b9ffac4320

https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3

https://github.com/Schachte/stable-matching-algorithm/blob/master/stable_matching.py

https://medium.com/predict/guide-to-the-6-best-electric-vehicle-charging-networks-703917ef374

https://medium.com/0xmachina/tesla-supercharging-fast-charging-ev-networks-f6077499bb9d

https://medium.com/extremetech-access/bmw-porsche-demo-super-fast-electric-car-charger-27ce7fd9695d

During production:

https://towardsdatascience.com/train-image-recognition-ai-with-5-lines-of-code-8ed0bdd8d9ba

https://realpython.com/python-web-applications/

https://medium.com/code-85/easily-build-your-first-web-app-with-pythons-flask-d12825f9f1d9

https://medium.com/educative/a-complete-guide-to-web-development-in-python-2a91dd518d0

https://stackoverflow.com/questions/54786145/web-cam-in-a-webpage-using-flask-and-python

https://towardsdatascience.com/camera-app-with-flask-and-opencv-bd147f6c0eec

https://towardsdatascience.com/wikipedia-api-for-python-241cfae09f1c

https://www.mulesoft.com/resources/api/what-is-an-api

https://pub.towardsai.net/tesseract-ocr-for-text-localisation-and-detection-e217a87f9d8d

https://betterprogramming.pub/building-a-photo-translator-using-python-with-the-google-translator-api-94b8da75f7f8

https://www.programmableweb.com/api/goodreads-feed-api https://github.com/mdzhang/goodreads-api-client-python

https://towardsdatascience.com/how-to-extract-text-from-images-with-python-db9b87fe432b

https://www.analyticsvidhya.com/blog/2020/09/integrating-machine-learning-into-web-applicatio
ns-with-flask/

https://blog.cambridgespark.com/deploying-a-machine-learning-model-to-the-web-725688b851c
7

https://towardsdatascience.com/building-a-web-application-to-deploy-machine-learning-models-
e224269c1331

https://medium.com/fintechexplained/flask-host-your-python-machine-learning-model-on-web-b5
98151886d

https://stackoverflow.com/questions/44926465/upload-image-in-flask/58232783

https://flask.palletsprojects.com/en/1.1.x/patterns/fileuploads/

https://www.geeksforgeeks.org/python-uploading-images-in-django/

https://stackoverflow.com/questions/44604405/django-uploading-image-using-builtin-user-model

https://medium.com/fintechexplained/flask-host-your-python-machine-learning-model-on-web-b5
98151886d

https://www.analyticsvidhya.com/blog/2020/04/how-to-deploy-machine-learning-model-flask/

https://stackoverflow.com/questions/4526273/what-does-enctype-multipart-form-data-mean

https://medium.com/@danishkhan.jamia/upload-data-using-multipart-16b54866f5bf

https://google.github.io/styleguide/docguide/best_practices.html

https://github.com/lschmiddey/book_recommender_voila/blob/master/App-with-Voila-mybinder.i
pynb

https://lschmiddey.github.io/fastpages_/2020/09/27/Prepare-Notebook-for-App-Part3.html

https://www.w3schools.com/css/css_howto.asp