DATE: February 9th, 2019

TO: Dr. Mary Eberlein

FROM: Team Eagle - Hiep Nguyen, Michael Flanders, Kunpeng Qin, Nabil Khan, Nelson Levy, Wesley Klock

SUBJECT: Team Eagles' BookBrain Project Proposal

This memo will describe the vision of our project, BookBrain, as well as other aspects of the web application's implementation and design considered throughout the software development life cycle. More specifically, we will describe our vision, users, use cases, requirements, competitor analyses, data management, tools, and feasibility.

VISION

The goal of BookBrain is simple: BookBrain will show users internet-aggregated ratings for books and provide book recommendations based on these ratings and genres. Usually when someone is trying to decide whether or not they want to read a book, they might do a Google search and be faced with many websites offering mixed reviews—one might display only one review with a 3/3 star rating and another might have hundreds of reviews with an average rating of 4.3 out of 5 stars. Furthermore, some of these sites might not even have reviews or ratings for books that just were just recently published. Our website would simplify the book recommendation experience for the user by compiling reviews from different merchant websites, social media feeds, and news feeds as well as providing trendy information about a book all on a single page. By doing this, we hope to give our users a more holistic review of the book based on overall internet reviews and trendiness.

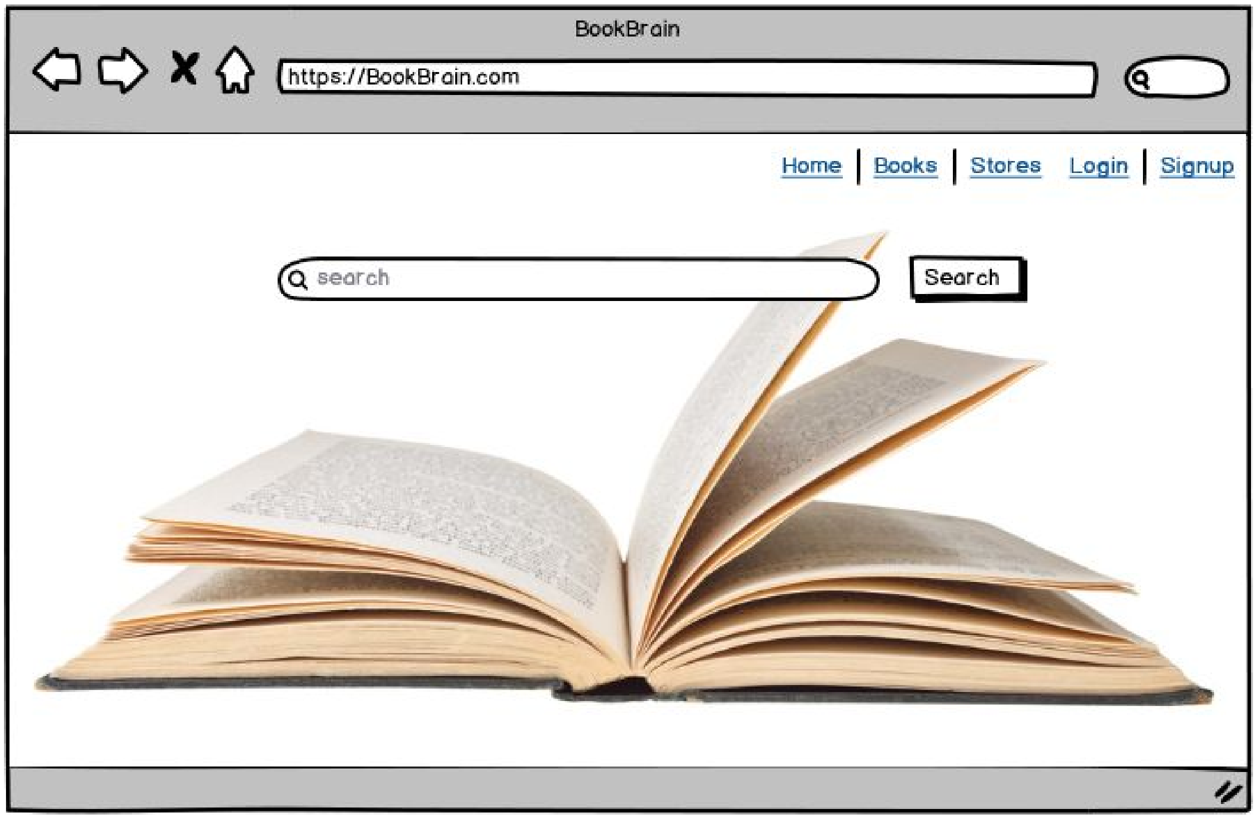
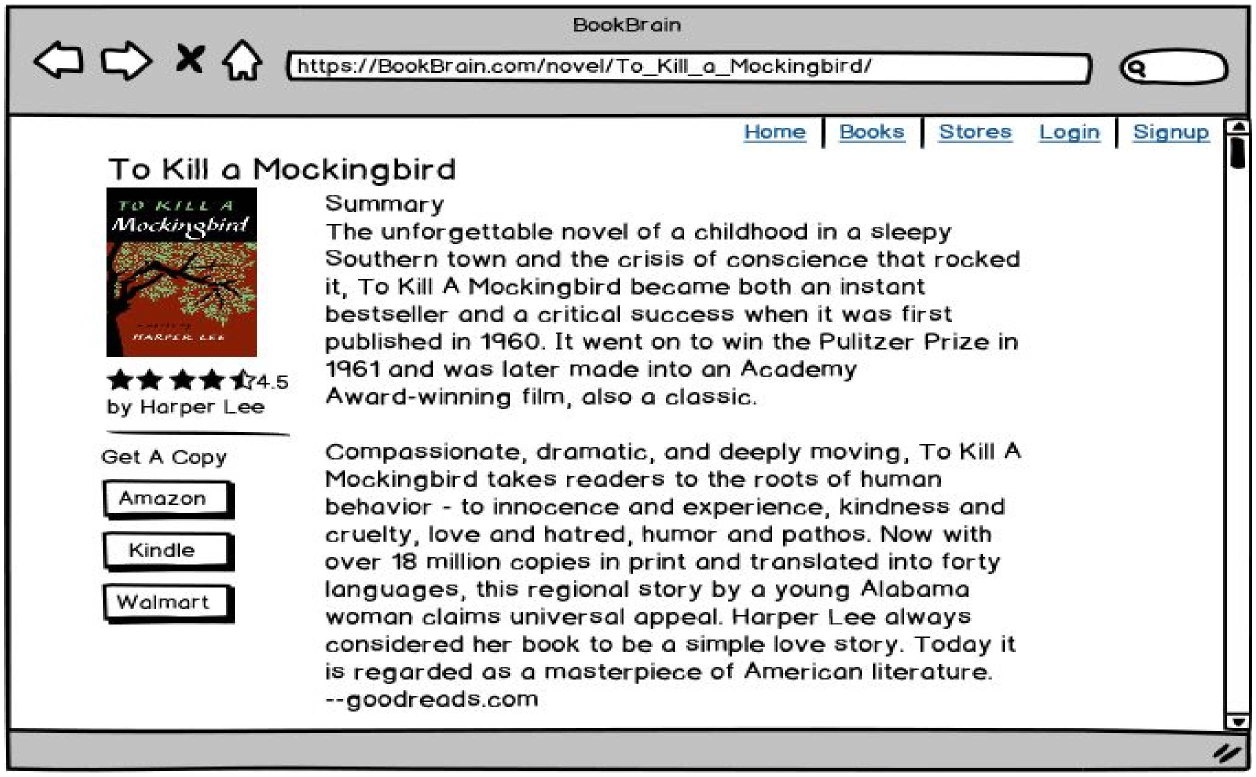


Figure 1. Wireframe of BookBrain’s Home Page (Designed on balsamiq.com)



To Kill a Mockingbird

4.5
by Harper Lee

Get A Copy

Summary
The unforgettable novel of a childhood in a sleepy Southern town and the crisis of conscience that rocked it, To Kill A Mockingbird became both an instant bestseller and a critical success when it was first published in 1960. It went on to win the Pulitzer Prize in 1961 and was later made into an Academy Award-winning film, also a classic.

Compassionate, dramatic, and deeply moving, To Kill A Mockingbird takes readers to the roots of human behavior - to innocence and experience, kindness and cruelty, love and hatred, humor and pathos. Now with over 18 million copies in print and translated into forty languages, this regional story by a young Alabama woman claims universal appeal. Harper Lee always considered her book to be a simple love story. Today it is regarded as a masterpiece of American literature.
--goodreads.com

Figure 2. Wireframe of A Single Book's Page (Designed on balsamiq.com)

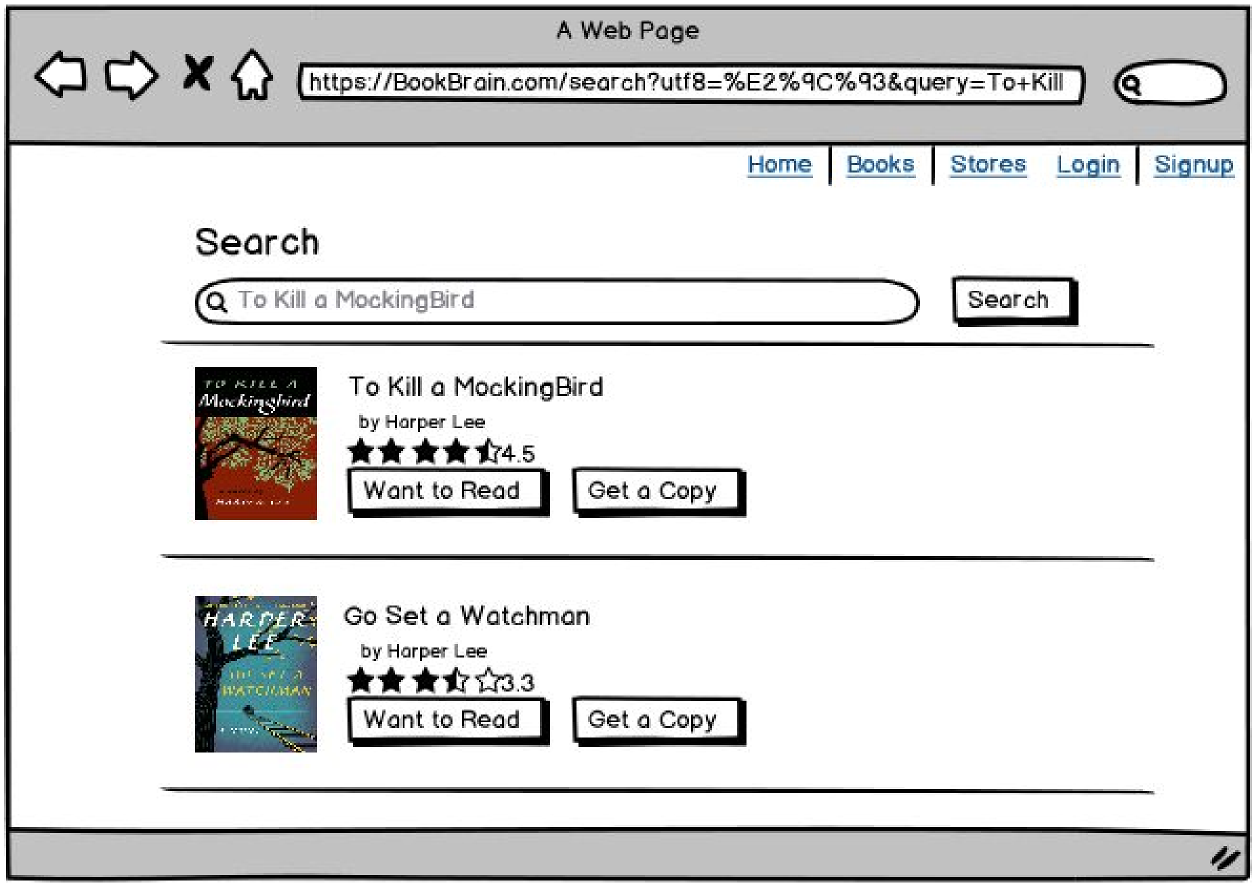


Figure 3. Wireframe of the Search Results Page (Designed on balsamiq.com)

USERS, USE CASES, AND REQUIREMENTS

The target demographic for BookBrain fall under two categories—someone that is looking for a new book to read and wants suggestions based off of genre and internet popularity or someone that is looking to purchase a specific book and would like ratings and reviews from many different internet sources aggregated in one place. To further explain our target user's wants and needs, we have 6 user stories, a use case diagram, and a formal use case that describe what information our target users are looking for when they visit BookBrain:

- As a person trying to determine whether or not I will like a certain book, I want to be able to search for a book and view an overall internet score of the book without visiting multiple pages.

- As a person trying to determine whether or not I will like a certain book, I want to be given a short description or summary of the book as a preview.
- As a person trying to find a new book to read, I want to be shown related books based on genre and an overall internet score of the book.
- As a person looking to purchase a book, I want to be provided links to purchase the book.
- As a person looking for book reviews, I want to be able to filter books based on genre and ratings
- As a person looking for book reviews, I want to be able to search for a specific book.

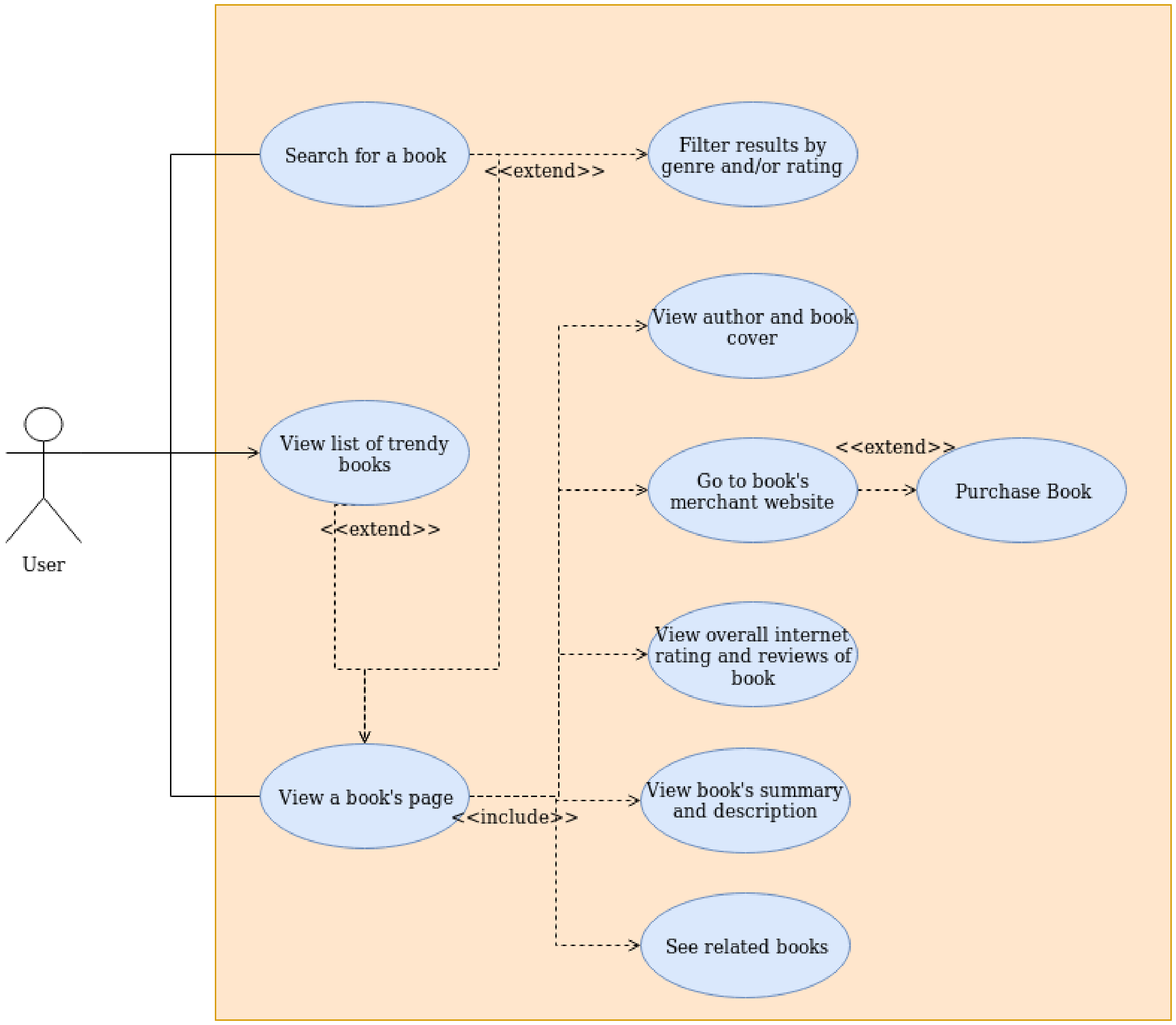

Search for a book
<<extend>>
Filter results by genre and/or rating
View author and book cover
User
View list of trendy books
<<extend>>
Go to book's merchant website
<<extend>>
Purchase Book
View overall internet rating and reviews of book
View a book's page
<<include>>
View book's summary and description
See related books

Figure 4. Use Case Diagram (Designed on draw.io)

Formal Use Case

- Use Case: Search for a book and view its page
- Primary Actor: User/Book reader
- Goal: User finds ratings and reviews for a book
- Trigger: User visits the website to look for a book
    1. User searches for a book.
    2. User views search results.
    3. User selects the book that (s)he was searching for.
    4. User views the book's page.
    5. User sees information for that book including: an overall internal rating, book cover, author, title, description of the book, other books related to this one by genre and rating, and a link to purchase the book.
- Extensions

  2a. The book does not exist and no similar results can be found.

    2a1. BookBrain informs the user that no results can be found.

    2a2. BookBrain suggests the user broaden their search.

  2b. User wishes to sort the results.

    2b1. User filters by genre or rating.

  5a. User wishes to purchase the book.

    5a1. User clicks on a link to a merchant's website to purchase the book.

COMPETITIVE ANALYSIS

Compared to other book rating websites, BookBrain is novel. It provides reviews from many different sources. In other words, BookBrain is more reliable than traditional, book-rating websites since it provides comprehensive review of the books. In the following paragraphs, we will look at book retailers such as Amazon and existing web scraping/sentiment analysis code, and give an in depth explanation on what they do and why we are unique.

Let us start with Amazon, one of the world's largest online marketplaces. Amazon provides book reviews written by the customers, however, many readers of the books may not want to put their reviews on Amazon. This could be due to laziness, forgetfulness, or some other factor. Thus, the book reviews on Amazon may be skewed to only a certain demographic of readers, giving a not-so-accurate rating. Our method for rating books avoids this issue by using data scraped from multiple web sources. Furthermore, when Amazon recommends books to its customers, it uses collaborative filtering to make its decision [6]. Collaborative filtering involves a lot of factors in making its decision, such as past purchases, cart history, and even items that were clicked on but never purchased [6]. This is great for general shopping needs, but to get a better recommendation for books, an algorithm may need to consider more specific factors. We are taking a different approach with our project by recommending related books by genre. When looking at our rating algorithm and recommendation procedure, it is clear that we can argue for the uniqueness of our project for not only Amazon, but also with most other online book retailers as well, such as Barnes & Noble.

Github repositories and other websites are filled with code from all over, and code for web scraping and sentiment analysis are no exception. Most online existing code for sentiment analysis lack one major component when providing a sentiment analysis for books. They all use existing book reviews to obtain a sentiment on how users feel. The issue with this approach is that not much new information is being portrayed, as the people can already tell at a high level whether someone liked a book or not based on the 5-star rating scale. Furthermore, most code bases online only scrape reviews from one website, such as either only GoodReads or only Amazon. Furthermore, we could not find any existing websites that provide a user interface (UI) for the general reader to actually interact with to receive a sentiment analysis for a book that they are interested in. BookBrain is much more unique in that we scrape text from unique sources such as online articles and social media platforms, not just existing reviews. BookBrain will also provide a UI that will be interactive and practical for the general reader. By not restricting ourselves to only one source of information, BookBrain will create a better, holistic approach to the book reviewing process.

DATA SOURCES AND STORAGE

We plan to use a variety of sources for our review data, including Amazon, Barnes and Noble, Google News, Reddit, and other sources that would provide valuable feedback on a book. We will store user information in a local database on our web server. Since the book review data needs to be updated dynamically we have opted for a cloud-based MySQL solution. The rest of this section will discuss a detailed implementation of source scraping and data storage.

Since information related to user accounts such as search history, saved books, login information, and comments will need to be updated constantly by the website, we decided to keep them close to home. A MySQL database will be hosted locally on our Flask site to provide the most up to date information. MySQL was chosen for its compatibility with cloud resources and its relational nature.

We plan to use a variety of sources to gather review data, including Amazon, Barnes and Noble, Google News, Reddit, and other sources that would provide valuable feedback on a book. The first step will be collecting the reviews. For Google News and Reddit, we will get relevant results through an application programming interface (API) request. For Google News, we will additionally run a hypertext transfer protocol (HTTP) request to pull the article referenced. For other sources, it gets a little more tricky. Since other sources dynamically load reviews using Javascript, we will have to use Selenium in Python to load the page and pull data using a scraper. An additional challenge we might face is internet protocol address (IP) blocking to detect scrapers [5]. We plan to combat this by using a common IP swapping practice for web scrapers. This process will be run frequently on an Amazon Web Services Elastic Compute Cloud (AWS EC2) instance to provide up to date information for the review database. This allows us to keep the response time of the site to a minimum by outsourcing a majority of the computing to a cloud resource.

AWS RDS MySQL system will be used host our review database. We chose cloud hosting instead of hosting locally because it allows us to dynamically update the review database without long website downtime. The website will interact with our database through API calls which are provided by AWS Lambda functions. We chose AWS instead of Google Cloud Platform for BookBrain primarily because of its compatibility with the rest of our architecture.

PLANNING AND SCHEDULING

We plan on splitting the development of BookBrain into four phases. The first two phases of development will be focused on learning new technologies and building a prototype of the web application and dynamic web content scraper. The main goal of the last two phases is refactoring, testing, and scaling the scraper to cover all of our sources. The rest of this section provides a detailed description and time breakdown of each phase as well as a phase chart.

During the first phase, we will focus on creating 14 static pages as a wireframe of our user interface and learning Python and Flask which serve as the foundation of our scraper and web application. We expect to finish phase one by February 25th, 2019 with about 12 days out of the 18 day period being spent on creating building a basic scraper and populating 14 static web pages each with content from a different book. The remainder of the time will be spent on documenting code and learning Python and Flask.

Phase two will last from the end of phase one until March 11th, 2019 lasting 14 days. During this time, we will focus on scaling the scraper, writing unit tests, and learning Selenium and React.js. We expect that the most time consuming part of phase two will be scaling the scraper as this will involve adding onto the scraper in a maintainable way, learning Selenium, and setting up the database; therefore, we have allotted 10 days out of the 14 day period to accomplishing these tasks. The other four days will be buffer room for writing unit tests for the scraper and learning React.js in preparation for the third phase.

Phase three is our longest phase and is dedicated to improving our user interface by adding dynamic web content such as filtering and sorting functionality and static and dynamic web page tests; we will also use this phase to add more sources to the scraper. This third phase should last from March 11th, 2019 through April 8th, 2019 totalling 28 days, and we expect that the bulk of the work will be adding more sources to the scraper, so we plan to spend the first 18 days on adding sources to the scraper and the last 10 days on adding dynamic web content.

The fourth and last phase will be from April 8, 2019 to April 29, 2019 totalling 21 days and will be focused on refactoring, fixing any errors, testing, and improving UI/UX. Because we plan on working on each of these tasks throughout the development process, we see this last phase mostly as refinement and bug fixes. As such, we expect to spend an even amount of time on each of these tasks.

Phase	Overview	Details
1	Design, Learning technologies, Basic implementation of the web application and scraper Deliverables: - 14 static web pages each with contents of a different book - Basic scraper with documentation	- Design and document the scraper and webapp - Add 14 static web pages that display data from specified in the use cases i.e. titles, book covers, ratings, reviews - Learn Python and Flask - Build basic scraping script and collect

		aggregated percentage based star rating for 14 books
2	Add ability for bulk scraping, Basic unit testing, learning technologies Deliverables: - Unit tests for scraper - Scaled-up scraper	- Add basic unit tests to scraper - Add ability to scrape in bulk (100-1000 books at a time at minimum) - Learn Selenium for dynamic web content scraping
3	Add more sources to scrapers, Add dynamic content to the site, Add tests for the web application Deliverables: - Scraper templates/code for all sources - Dynamic web pages with filtering and searching functionality - Tests for web pages	- Add social media sites (e.g. Twitter, Reddit, etc.) and news feeds to scraper sources - Learn React - Add filtering/searching functionality to the site - Add tests for static/dynamic web pages

4	Fix any lingering errors, Refactor, Continue to evaluate test coverage and write more tests, Improve UI/UX Deliverables: - Refactored code with design patterns added - Project report - More tests if needed - Fully functioning web application	- Refactor and add design patterns - Evaluate test coverage and write more tests - Produce an overall project report - Improve UI/UX - Work on final presentation

TOOLS, SOFTWARE, AND FRAMEWORKS

The development of BookBrain will be broken down into many small subtasks which require learning and implementing programming languages, frameworks, and libraries. These subtasks will include scraping websites, maintaining a database, creating and hosting a server, creating a front-end webpage, and determining the sentiment of scraped reviews from varying websites. In the following paragraphs, we will describe which tools, software, and frameworks we plan on using to accomplish these subtasks.

The core of our project revolves around the idea of using other websites, articles, and social media platforms to create ratings for various books. We will use Selenium, a web browser automation tool, to scrape important text we need from these sources. Selenium provides an API for Python [3], which is an easy to pick-up language. Furthermore, there are a good amount of tutorials online that demonstrate how to effectively scrape websites with Selenium. Learning Selenium will be one of the first things we do in phase two, because it is essential in scraping paginated reviews. We plan on storing the data that we scrape with Selenium in an MySQL AWS RDS instance. Depending on the amount of data we scrape and the execution time of our scraper, we will decide which database best fits our needs by the end of phase two.

While BookBrain's code will be hosted on Github (https://github.com/Flandini/461L-Eagles), our running web application will be hosted on an AWS EC2 server, so it is important that we make it as robust and modular as possible. We plan to utilize Flask as the framework for our server. Flask is implemented in Python [2], so we do not need to learn another language for this subtask. Also, since we may need to use machine learning to analyze the sentiment of reviews, using a framework made for Python allows for easier integration of any Python machine learning code. For testing server responses and behavior, we plan to use Postman to make simple requests to the server. Postman will be essential in the beginning phases of our design, because it will allow us to make API requests while we build up the actual front-end portion of the website. With Flask and Postman, we hope to create a effective and reliable server for our project.

The front-end webpage will have both static and dynamic portions, and we felt the most efficient way to construct our website would be through the React framework for Javascript. React is great because unlike other front-end frameworks, it breaks up the UI into components, allowing for more flexibility and complexity [1]. For example, if one portion of our webpage is static, but another requires an update every few milliseconds, React can handle this effectively by creating two separate components for the sections in the website. By retaining the states of its components, React makes it easy to keep up with data changes and keeps things simple when creating a UI for a webpage [1].

The last major piece of technology we plan to use is TextBlob, a "Python library for processing textual data" [3]. We will use TextBlob to determine the overall sentiment of a person's review based on the person's choice of words. Since TextBlob is callable from Python just like Selenium and Flask, it makes integration and implementation that much easier. We will aggregate the results we obtain from TextBlob to present an overall internet rating to our users.

For our project, we plan to use Selenium to scrape websites for data, either an AWS S3 bucket or MySQL for our database, Flask as our web application framework, Postman to test the server, React for our front-end UI, and TextBlob for sentiment analysis. When choosing these tools and frameworks, we considered technologies that would integrate seamlessly, allow fast-paced development, and scale to meet our large data needs.

FEASIBILITY

Although the idea of building a website that provides comprehensive book reviews is feasible, there are many challenges that BookBrain will face as a 'smarter' book review site such as scraping of dynamic web content, formatting the scraper for different websites, and rate limiting of our scraper. When researching our competitors, we could not find another book review website that pulls reviews from multiple sites or that presents users with an overall-internet score. We believe that this is due to the difficulty of scraping reviews from multiple sites, news feeds, and social media feeds. Many of the reviews or articles in these feeds are paginated and require a dynamic, web-content scraper which is slightly more time consuming and complicated than a static web page scraper.

We'll also need to spend a large portion of our development time on formatting the scraper for different websites and avoiding rate limiting. As outlined in the planning and scheduling portion of our proposal, we are projecting that the largest time sink will be adding all of our sources to our scraper. Our scraper for Reddit will have to traverse a much different document object model than our scraper for Amazon or Barnes and Noble. Furthermore, there are already resources available for scraping Amazon reviews, but a quick Google search shows that there are no review scrapers for our other sources such as Goodreads.com or Barnes and Noble. Consequently, we will probably have to dig through the HTML of a few reviews for each source we want to add in order to have the scraper pull the information we are interested in.

Another difficulty that we will face is that our project is very dependent on having a large amount of scraped data; this means that we will need to either start scraping early and run for a long time or build our scraper to be very quick. However, almost everyone that has tried the latter has faced the dreaded rate limit. For example, a large website such as Amazon can almost definitely track web crawlers, so we will need to find a way to overcome rate limiting whether that means using a prebuilt scraping platform or an HTTP proxy server [5].

We have brainstormed the difficulties that we will face when developing BookBrain, and we decided that the above three points will be the largest obstacles we face. Since our project relies heavily on scraping and analyzing text data from many different websites, we believe that keeping these obstacles in mind while diving into the development of BookBrain will help us allocate more time towards these more difficult tasks.

CONCLUSION

In conclusion, our project, BookBrain, would provide a new way of exploring books for book readers by presenting them with data such as overall internet ratings, summaries and descriptions, related books based on rating and genre, and links to purchase the book all in one page. We will accomplish this by using tools such as a reliable cloud database, an efficient web scraper, and a user-friendly interface. Overall, we believe the success of this project would contribute greatly to the book reading community around the world.

REFERENCES

[1] Facebook Incorporated, “React – A JavaScript library for building user interfaces,” *ReactJS*. [Online]. Available: https://reactjs.org/. [Accessed: 06-Feb-2019].

[2] GitHub. (2019). *pallets/flask*. [online] Available at: https://github.com/pallets/flask [Accessed 8 Feb. 2019].

[3] R. Taracha, “Introduction to Web Scraping using Selenium,” *Medium*, 04-Sep-2017. [Online]. Available :https://medium.com/the-andela-way/introduction-to-web-scraping-using-selenium-7ec377a8cf72. [Accessed: 06-Feb-2019].

[4] S. Loria, “TextBlob: Simplified Text Processing,” *TextBlob: Simplified Text Processing - TextBlob 0.15.1 documentation*, 2018. [Online]. Available: https://textblob.readthedocs.io/en/dev. [Accessed: 06-Feb-2019].

[5] Brody, H. (2016). *How to Scrape Amazon.com: 19 Lessons I Learned While Crawling 1MM+ Product Listings*. [online] Blog.hartleybrody.com. Available at: https://blog.hartleybrody.com/scrape-amazon/ [Accessed 9 Feb. 2019].

[6] Rejoiner, “Amazon's Recommendation Engine: The Secret To Selling More Online,” *Rejoiner*, 28-Aug-2018. [Online]. Available: http://rejoiner.com/resources/amazon -recommendations-secret-selling-online/. [Accessed: 09-Feb-2019].