STEP Pod Project Design Doc Template

Authors: <[Danya Gao](mailto:danyagao@google.com), [Ankita Mitra](mailto:mitraan@google.com), [Adrian Manhey](mailto:amanhey@google.com)>

Reviewers: <add yourselves here>

Status: Draft  
Originally Proposed: 2020-06-16 / Last Updated: 2020-06-17

*See if you can keep this document shorter than 5 - 8 pages. Depending on diagrams and tables, that may not happen, and that’s fine.  
Italic text is intended as instructions that can be deleted. These two lines should be deleted, as well.*

# Brainstorming

-authentication, books, Youtube, blobstore image upload, google cloud vision, charts, sentiment analysis(?), google news

* P0 (PROTOTYPE, finish by week 6)
  + Books API
  + Home page with all books, and be able to loading different book pages
* P1 (MVP, finish by week 8)
  + Comments
  + Authentication
  + Sentiment Analysis (possibly using the “common words” part of google books)
    - Helps recommend books, sad book = sad background
* P2 (Week 9 - end)
  + Blobstore/ Google Buckets / Google Cloud Storage
    - Help with book recommendations
    - People upload their own writing
  + Helpful youtube links
    - Literature analysis videos (thug notes, crash course)
    - Audio book links
  + Google News
    - Controversies, updates/new releases from that author
  + Charts
    - For everything

# Objective (adrian)

*(1-2 paragraphs) What are we doing, why, and for whom?*

*What problem are you trying to solve? Consider the goals and NON-goals, and make the objective understandable for someone unfamiliar with this project. Elaborate on the details below in the Background and Requirements sections.*

The problem is that there is no review-focused resource for books online. We are creating a webapp so that users can leave reviews of books, with more emphasis on the content of the reviews than a star-rating system. The goal is to have a resource where people can express their pleasure or displeasure with different pieces of literature and find books that they might enjoy based on what they have already read. The recommendation system will rely on more of the sentiment of the text rather than the genre, so readers can find books with a similar feel. It is not our goal to create a superficial system of recommendations, but a place where passionate readers can relay their experience with different books.

# Background (adrian)

*(1-2 paragraphs) What background context is necessary?*

*You should mention related work that you’ve done, if any. Also, try to justify the need for this with a compelling user story; user-centered design is an important part of the design process. If you have any supplementary documents, please include document titles so they can be found when links go bad. Include a glossary if necessary. Note: this is background; do not write about your design, specific requirements details, or ideas to solve problems here.*

In high school while trying to find good books on recommendation sites I never found exactly what I needed either because they were trying to sell me something or the recommended books were only recommended from same/similar genres and authors. If there is a way to recommend books based on the book itself, not solely relying on the author or the genre, we believe it would yield more enjoyable recommendations. We believe we can begin to accomplish recommending books based on their “vibe” by using sentiment analysis.

# Requirements and Scale (danya)

*(2-5 paragraphs) What needs to be done?*

*What problem are you trying to solve? Who are the customers for this solution? Your job here is to quickly educate others about the details you know about the problem space, so they can help review your implementation. Roughly estimate relevant details. How big is the data? What are the transaction rates? How many computers are involved? Bandwidth? Growth of data or traffic? If you don’t know the answers to these questions, ask your podmates or hosts and make an educated estimation. It’s also okay to not answer these questions if they’re not relevant to your app.*

*Remember, the better and more succinct your summary is, the better others can (a) understand the problem and your design, (b) help you consider alternate designs, (c) help you refine your ideas.*

The problem we are trying to solve is that there doesn’t exist a review-focused book website that consolidates resources from the internet. In addition, book websites generally only make recommendations based on genre, rather than the sentiment that book evokes, which is often an important indicator of how likely it is that someone will enjoy a book. Book websites focus on either the social component (e.g. “Your friends have been reading this!”), the cataloging component (e.g. “These are the books you’ve read”), or the selling component, but rarely the review and recommendation component, which can be immensely helpful to those that want to evaluate which books are best for them.

The customers for this solution would include students and teachers, who need recommendations on books to read and resources on book analysis, and the wider community of book readers, like book bloggers. While there is no concrete data for the amount of book blogs that exist, a google search for “book blog” turns up page after page of results like “top 20 book blogs you need to be following” and “how to start a book blog”, so we know that there are many people invested in the review of books, and looking to review books themselves.

There is a large internet community focused on book reviews; for example, as of July 2019, Goodreads has 90 million registered members, growing at about 5 million per year. As of October 2018, they had 80 millions reviews for 2.3 billion books, 50 million monthly visitors, and 430 monthly pageviews. Book review and cataloging site LibraryThing had 2.4 million users and 135 million books as of June 2019.

Also, as part of our design, we plan to incorporate the Youtube Search API, for analysis of literature, as well as to allow people to listen to audiobooks if they are unable to read or access the text of a book. From 2016 to 2017, consumption of audiobooks increased by 28.8%, making them the fastest growing book format. Audiobooks are primarily used by those under the age of 35, with 48% of users falling into this category. They are also ideal for commuters with limited free time, with 32% of users listening in the car.

If we have time, we also plan to incorporate file upload functionality, so that users can self upload their own writing. In 2017, over 1 million books were self published in electronic/print form.

# Design Ideas (ankita)

Landing page:

The landing (main) page will have a scrollable grid of book titles (image covers) that can be clicked on to choose a title. We will also give users the option to pick a title using a dropdown list (text of title). This will be largely implemented using HTML/CSS and Javascript. When a title is chosen, we reach a page specific to that book. We plan to use a Book.java object and a large dataset of books. We limit ourselves to this set instead of allowing users to search for any book, as this will allow us to implement our own recommendation algorithm if we have time.

Book-Specific Pages:

User Contributions:

* The user will be able to leave their comments/short reviews on the title. We will implement the storage and retrieval of comments using **Datastore\*** with Servlets. HTML/CSS will be used to create *form* objects that the user can interact with to leave comments.
  + TODO: test the [Vision AI API](https://cloud.google.com/vision)
  + If it works, we will allow users to upload pictures that remind them of the book/fanart and use properties from the image for sentiment analysis.
* Users will also be able to upload any fanfiction/in-depth reviews/papers they’ve written based on the book. We will be using **Datastore\*** to store and retrieve these uploaded files. Since these files may contain private information, we will require authentication before a user can upload files. Authentication will be done via [Google Sign-In](https://cloud.google.com/appengine/docs/standard/java/oauth#google_sign-in) since the Users API is [no longer recommended](https://cloud.google.com/appengine/docs/standard/java/users/).
* We will be using both forms of user contributions to improve our sentiment analysis scores using the [Cloud Natural Language API](https://cloud.google.com/natural-language/docs/classifying-text)^ (described in the next section).

***\*Datastore*** *seems to be “no longer recommended”, so we will probably try out* [*Cloud Storage*](https://cloud.google.com/appengine/docs/standard/java/using-cloud-storage) *instead.****^****The Natural Language API will give us multiple dictionaries of {category name, confidence}. We will use these sets and our own algorithm to come up with a final normalized dictionary representing the sentiment of a title.*

API Contributions:

* [Books API:](https://developers.google.com/books/docs/overview) 
  + We will use this to get information about the top result of the *query = bookTitle* and then display this information on the page using the servlet and HTML.
  + Use the preview/description text as well as the genre of the top 5 results of the query to use with the Cloud Natural Language API (details below).
  + TODO: Check if the “common words” section can be retrieved using the API.
    - If so, we will also use this to contribute to our final sentiment analysis algorithm.
* [Youtube Search API](https://developers.google.com/youtube/v3/docs/search/list):
  + We will use this to get the top result for the *query = bookTitle + “audiobook”*  
    By embedding this top video using HTML and the Java Servlet that uses the API, we should have a playable audiobook for the selected title. Since this may not work 100% of the time, we will have a button/form that allows the user to mention that the result is incorrect/not relevant.
  + We will also use this to get the top 5 results for the *query = bookTitle + “analysis”* and will receive user feedback in the same way as above.
* [News API](https://newsapi.org/):
  + We will use this and pass in *query = bookTitle* to get the top 2 results, and   
    *query = author* to get the top 2 results, and then display them using the Java Servlet and HTML. In addition, we will use the form mentioned above to receive user feedback.
* [Cloud Natural Language API](https://cloud.google.com/natural-language/docs/basics):
  + We will use the Content Classification service of the API. This returns multiple dictionaries of {category name, confidence} when provided a certain user input. The inputs we plan to use are:
    - Preview text (if available) of the book
    - Book description
    - Book genre
    - “Common Words” (if possible)
    - User Contributions (described in previous section)
  + We then create our own algorithm to use the results from all these function calls and combine/normalize them in a way that makes sense, so that we have a final dictionary of {category name, confidence}
    - Using this algorithm, we can create charts as described below.
    - We can also find books that have the most similar dictionary and recommend those to people who are on this page.
* [Charts API](https://developers.google.com/chart)
  + We will create a pie chart with the most common category names to display the general “feel” of the book. For instance, this could be a pie chart with ⅓ Murder, ½ Comedy, ½ Family.
  + We will use a line chart to display the number of times people have contributed to this page over time.

From the book-specific pages, there will be a button to go back to the home page.

# Timeline (adrian)

*(Filled table) Who is responsible for what, and when will it be done by?*

*Now that you’ve expanded on your implementation ideas, list out which of your podmates is going to be responsible for which parts of the implementation and when those should be done by. Ideally, this allows you to reference this document as you’re developing (and your hosts can reference it during 1:1s and your final evaluation).*

*This* ***absolutely*** *should not include code or diagrams whatsoever; simply name the thing that you expect to develop, add the due date, and add the owner or owners. Make sure to be granular; you cannot simply put the last week of your internship and “Complete project” as a task. Having smaller features due by specific dates will help your hosts ensure that the project is being scoped appropriately, and it will help you remain accountable to your podmates and to your hosts. It’s possible for more than one person to own a line in this table, but if that happens, consider breaking that up into multiple tasks that can be owned independently (unless you plan to pair program the task).*

*Feedback that you get on this section will mostly be on how you’ve broken up the work for this project. It should be fairly even, since this is a team effort, and this timeline will help ensure that you’ve broken up the work in a good way. You may also get feedback asking for features to be cut for time or added to make sure that you’re challenging yourselves. When you receive the feedback, you should incorporate it into this document by updating the table below as needed or moving ideas out of the table and into Alternatives Considered.*

***Example:***

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Due Date** | **Deliverable** | **Owner(s)** |
| Phase 0:  Prototype  Due: 06/26/2020 | *04/20/2020* | Complete design document template. | eyurko |
|  |  |  |
|  |  |  |
|  | *04/30/2020* | Review design document feedback; incorporate it into next draft. | eyurko |

# Alternatives Considered (danya)

*(2 paragraphs) Include alternate design ideas here which you are leaning away from and short justifications as to why.*

One alternative design idea we considered included allowing the user to search for any book they want, and be able to get reviews/recommendations for that book. We decided against this because it’s outside the scope of our current project, and it will be easier to focus on book recommendations/gathering resources with a predetermined set of books. If we were to expand this project, we could allow for a much larger set of books.

Another alternative design idea is adding unreleased books to our database, such as hyped up sequels like 2015’s *Go Set a Watchman* (a “sequel” to *To Kill a Mockingbird*). However, we are deciding against that because a main purpose of adding unreleased books to book review sites is to get preorders and sell more books, which is not the intention of our application. In addition, one of the biggest complaints of websites like Goodreads is that they allow users to leave reviews before books are even released, which means that reviews are oftentimes based on excitement or preconceived notions, rather than the books itself.

06/17/2020

Feedback:

* focus on a couple things that we want to focus matching on
* need to find a way to do offline processing (e.g. tagging, labeling) so that the results are ready before the request arrives.
  + have a certain amount of books pre-processed and match with those
  + cannot operate in a timely manner if all done at the same time
  + processing independent of the user request; can do this in the cloud with compute instances

Next Steps

* List the independent work items/features we want (e.g. functionality, pages)
  + Use-case diagrams (flow-charts/wire-framing) help to visualize what the user can do