Software Engineering Proposal – Spring 2019

Group Members:
Vedanta Dhobley: vjd41@scarletmail.rutgers.edu
Avani Bhardwaj: ab1572@scarletmail.rutgers.edu
Shazidul Islam: si194@scarletmail.rutgers.edu
Akshat Shah: avs91@scarletmail.rutgers.edu
Kutay Kerimoglu: kk851@scarletmail.rutgers.edu
Alan Patel: akp122@scarletmail.rutgers.edu
Anthony Matos: amm720@scarletmail.rutgers.edu
Joel Cruz: jc2125@scarletmail.rutgers.edu

Project Title: Recommending Alpha
Group Number: 16
Idea Description: A recommendation algorithm will be made that inputs a list of books and uses tags (e.g. author, genre, age/demographic, etc.) to return a list of books which will be ranked (most likely using thresholds).
URL of our projects web-site:
Github Link: https://github.com/vedantadhobley/SoftwareEngineeringProject2019
We will be using this repository to commit all changes to our project. All form of communication will be through weekly group meetings, messages, and smaller group meetings where smaller groups of people will meet to finish mini-projects.
Team Profile:

a. **Individual Qualifications and Strengths:**
    Avani: C++, data organization, documentation, presentation, management
    Vedanta: Java, Python, AWS, design, presentation, management
    Shazidul: C++, Java, Python, SQL, design, management
    Alan: Java, Python, SQL, design, data analysis
    Kutay: C++, some Java, presentation, programming error checking
    Anthony: Java, C++, presentation
    Joel: C++, presentation
    Akshat: C++, Java, Python, SQL, design

b. **Team Leader:** Vedanta Dhobley

Problem Diagnosis:

A current issue at hand is that most "Recommendation" programs for books simply search with only author or genre tags when looking for books at libraries or bookstores. This leads to titles being recommended that are not similar to the person's liking or all recommendations which have one very odd tag match but don't have any other similarities to the first.

a. Example: If someone were to say they read a disney book, what can happen is that the recommendation algorithm will search for similar genre or author. However a better search would be Disney tag. Disney would return more suitable recommendations for the person that read something that was Disney related.
b. Another Example: Many who put that they have read Harry Potter, seem to get back a recommendation of Twilight due to it falling under the fantasy genre. However a series such as Percy Jackson would be a better match due to its use of magic along with fantasy, which is a more precious and intuitive approach.

The target for this program would be to entice avid readers that have issues with finding books similar to their taste, or those that simply do, can't decide that their taste is. With this algorithm, these readers can narrow their searches and find material that is much more relatable to the genre of books they have interest in. There is no age target since everyone at any age can enjoy reading!

Proposed Solution:

In order to provide a system that can implement better search results, we want to create an interface that allows the user to find reading material that is closest to their interest level instead of them relying on current recommendations that are only similar in one or two ways. As mentioned before, a good example of a bad recommendation is being recommended to read 'Twilight' or 'Lord of the Rings' to someone who wanted to read something much more relatable to 'Harry Potter.' The most general similarity between these three books is that they all fall under the 'fantasy' genre. Other than that, there is probably little to no similarity between these three series at all. Our goal as a team is to identify more specific details about books and give recommendations for books that are much more similar in taste. 'Percy Jackson' would be a relatively good recommendation for someone who has read 'Harry Potter' because it also applies the concept of magic being a major factor in both series.

In order to implement our solution, we can create more tags that can be added to books in terms of what key features each book has. Each tag will have a set multiplier (1x, 5x, 10x, etc) added to the books. The age tag will have a x50 or so multiplier to ensure that the recommendations that are coming back are in the same age limit of the user/reader. We definitely would not want to recommend books such as 'Lord of the Rings' to a five-year old. To see if a book fits well with the user's search selection, we will check how many tags match up between the two books and then add the multipliers together.

After picking the top twenty or so books, we will break them down individually into specific tiers. There will be 4 tiers total: Tier S, Tier A, Tier B, and Tier C. Tier S will be the one with the closest match to the recommended search and Tier C will be the least close, but still somewhat related to, the book that is being searched. Based on a statistical analysis of the results and using a standard deviation to split the tiers, we will decide a baseline score required for each book to earn a spot on the list of recommended books.

To optimize search results as fast as we can, we want recommendations to start showing up as soon as the user is searching up a book. To do that, we thought about doing a drop down menu as the user searched for a book but we decided that implementing this feature would result in a drop down menu so large it would hinder the user, rather than create efficiency. So ultimately, we decided to focus on predictive search completion instead. We want the search bar to predict the name of the book that the user is inputting, in order for it to find an actual match rather than trying to find nonsensical words or books that have been misspelled because these would not show up in the database at all. For the database, we are thinking of using SQL to store and search book tags. The idea is to make a very large database that we update about once a week. Other than that, the search would be more local because calling an API a ridiculous amount of times for every search would not work out too well.

The typical customers for our proposed system would be people who take an avid interest in reading or for parents who want to choose the best books for their children to read. Many people spend a lot of time trying to find books that match their reading interests but are disappointed after reading books that were recommended to them. Our goal is to ensure that more people are satisfied with their recommended books and continue to use our algorithm for future reading materials.

Functional Features:

- Ability to input an infinite number of book entries
- Using BFS to optimize run time
    - Start with looking at the tag that has the highest point value and then from there start looking deeper into the lower point tags to get a more accurate match.
- Use point system to rank recommended outcomes
    - The point system will help give a more accurate recommendation and be a deciding factor between two books that were close but one say had a better fit for the user
- Take the highest score and use that to create percentage brackets to categorize recommendations in S, A, B, C tiers

Plan of Work:

a. **Functionality:**
   - Tag Comparisons with all the other books that are in the database already.
   - The storage of the results that were pulled.
   - Categorizing them into tiers via percentile.
   - By doing multiple tries using the algorithm, we can detect how the results are compared to doing the search without the algorithm.

b. **Qualitative Property:**
   - We are going to be creating a website that the customer will be using, displaying a use of intuitive UI (User Interface). As the customer searches his or her book, the predictive text element of our interface will allow the customer to select the book much quicker.
   - We are going to be using a Raspberry Pi to have a server running at all time
   - When listing the books, the tier will give a quick description about what is similar to the books that user input and why it would be a good read.