CS 3354 Software Engineering
Final Project Deliverable 2

Bound 2 Be

Group 5: Charles Bacani, Victoria DeLozier, Peter Hoang,
Kim Le, Daniel Liu, Kim-Nhi Ngo, and Pranay Yadav

Delegation of Tasks

- **Charles Bacani**: Created functional requirements, calculated Function Point algorithm estimates and pushed deliverable 2 report to project GitHub repository
- **Victoria DeLozier**: Created Architectural Design diagram, project scheduling and conclusion
- **Peter Hoang**: Estimated cost of software, created nonfunctional requirements, edited and published presentation video
- **Kim Le**: Designed sequence diagrams, comparisons of similar work, references, and outro
- **Daniel Liu**: Designed Use Case Diagram, estimated cost of hardware, pushed presentation slides to github, presented cost slides
- **Kim-Nhi Ngo**: Pushed project scope to GitHub repository, identified and described software process model, presented project objective and project schedule
- **Pranay Yadav**: Initialized GitHub repository, designed class diagram, wrote unit test code, determined test cases & expected results, pushed code to project GitHub repository, designed concept for User Interface

Presentation Method

Our team decided to choose option 2 for our presentation mode. Our recorded, captioned presentation can be found at this URL:
https://web.microsoftstream.com/video/0edf8f52-8b38-4684-a5f7-911d204f0594?list=studio

GitHub Repository

Our project GitHub repository contains our report material and unit test code. It can be accessed here: https://github.com/pranay-yadav/3354-Bound2Be

Project Deliverable 1 Content

Addressing Proposal Feedback

Final Project Draft Description:

Group 5
Project Title
Bound 2 Be
Group Members
Charles Bacani, Victoria DeLozier, Peter Hoang, Kim Le, Daniel Liu, Kim-Nhi Ngo, and Pranay Yadav
Project Description
A book matching software where users can enter their interests and preferences to receive book recommendations and find places to purchase those books.
Motivation
We want to make it feasible to find books that match people's interests, fostering a love for reading and learning that will last a lifetime. This software could be used by book subscription services so they could provide books to their users that are catered to their individual preferences. Libraries and bookstores could also use it if someone asks for a book recommendation. The software can also be used by individuals who read for fun.
Tasks

- Charles Bacani
  - Management of project costs and pricing
  - Make an evaluation of work to conclude project
- Victoria DeLozier
  - Responsible for submitting each stage of the project to eLearning
  - Edit and put together presentation video
- Peter Hoang
  - Listing software requirements
  - Project scheduling
- Kim Le
  - Defining project scope and mapping out the general structure of the project.
- Daniel Liu
  - Create diagrams (case diagram, sequence diagram, class diagram).
  - Architectural design of the project.
  - Create the test plan.
- Kim-Nhi Ngo
  - Set up and manage Github repository
  - Create presentation on Canva
- Pranay Yadav
  - Help with creating diagrams
  - Comparison of similar designs

Presentation preference
Option 2: Video presentation with captions
Interested in scholarly paper?
We would not want to write a paper on this project.

Instructor Feedback:

> “A practical idea of a tool that promises a lot of potential use. In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how. Fair delegation of tasks. Please share this feedback with your group members. You are good to go. Have fun with the project and hope everyone enjoys the collaboration."

Response:

The main component of the feedback that we received as shown in Table 2 was that we need to be sure to include any similar applications and how Bound2Be is different in the final report. We have determined that there are a fair number of similar services, the main ones being GoodReads and Tailored Book Recommendations (TBR). When a user sets up an account with GoodReads, they are asked for their preferences but the questionnaire doesn’t go very in-depth, and as far as we could tell, is not dynamic as the user’s preferences change. TBR is a subscription-based service, providing 3 recommendations to the users every 3 months. When designing Bound2Be, we made sure to implement both a dynamic recommendation algorithm and an in-depth preferences questionnaire.

Software Process Model

The project employs the waterfall process model in the development process, primarily because the project is heavily plan-driven and encourages documentation. Our team meets regularly to plan, delegate, and schedule our tasks and activities before starting development. In addition, by the end of each phase of the project, documents are produced and approved by a passing grade being given and prior documents are modified accordingly based on changes within our project. Much like how problems from previous stages become evident in future stages in the software development process when using a waterfall process model, our team has discovered a few new issues as the project develops and changes must be made.

Software Requirements

Functional

1. A user shall be able to easily find the links that are associated with the books they want to read more about and/or potentially purchase.
2. The system shall generate each login, for each user, a list of recommended books that match the preferences of the users if they are not already check marked as read.11
3. Each user using the system shall be uniquely identified by their username and userID.
4. The system must allow users to register and log in into their accounts by entering their email and password.
5. The system must allow users to change their preferences at any time and update accordingly while also allowing users to change their list of favorite books.
6. The system must allow users to delete their accounts and associated information with ease.

Non-Functional

1. **Usability requirement:** 80% of users trying the application for the first time should be able to find at least 5 book recommendations within 5 minutes of software first time setup.
2. **Performance requirement:** The application should ensure that recommendations based on user input are matched, fetched from the database, and displayed within 1 second of system login.
3. **Space requirement:** The software will require no more than 5GB of storage on desktop systems to operate, including after subsequent updates.
4. **Dependability requirement:** The software will be down, unavailable, or under maintenance for less than 2% of operating and online time.
5. **Security requirement**: The system will support cloud-based authentication of user login credentials.
6. **Environmental requirement**: The software will run in various system environments, including macOS X or greater, Windows 8 or greater, and Linux and Unix based systems, as well as mobile platforms including iOS 12 or greater and Android 10 or greater.
7. **Operational requirement**: The software will be used by either individuals or libraries to recommend books based on given input and will be available as a stand-alone application for online download.
8. **Development requirement**: The software will be developed using JavaScript, and the MySQL framework will be used for interfacing with the database. The user interface will be developed using React.js and mobile implementations can be developed with React Native.

9. **Accounting requirement:** The system will track revenue generated through user purchases through referral links for books and/or advertisements and provide a monthly statement of expenses on the last Saturday of each month for accounting purposes as required by law.
10. **Ethical requirement:** The system will only use user data for book recommendations and diagnostic purposes. User data will not be sold or distributed and will be kept entirely confidential.
11. **Safety/security requirement:** The system will support database encryption and will not provide administrative accounts with more access than needed to prevent data leaks and security compromises.
12. **Regulatory requirement:** The software will adhere to a privacy policy informing users of the storage and collection of their account details and usage data in compliance with state data privacy laws such as the California Consumer Privacy Act (CCPA) and California Privacy Rights Act (CPRA).

Use Case Diagram

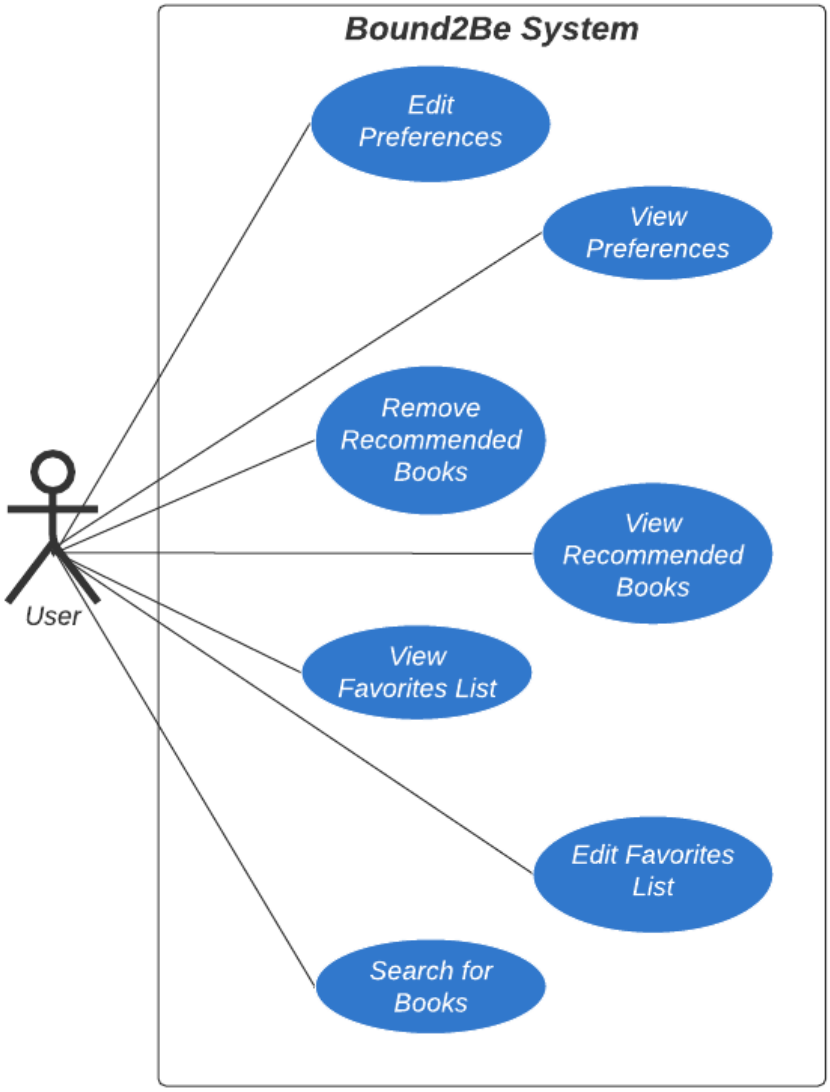

Figure 1. Use Case Diagram

Sequence Diagrams

Edit Preferences

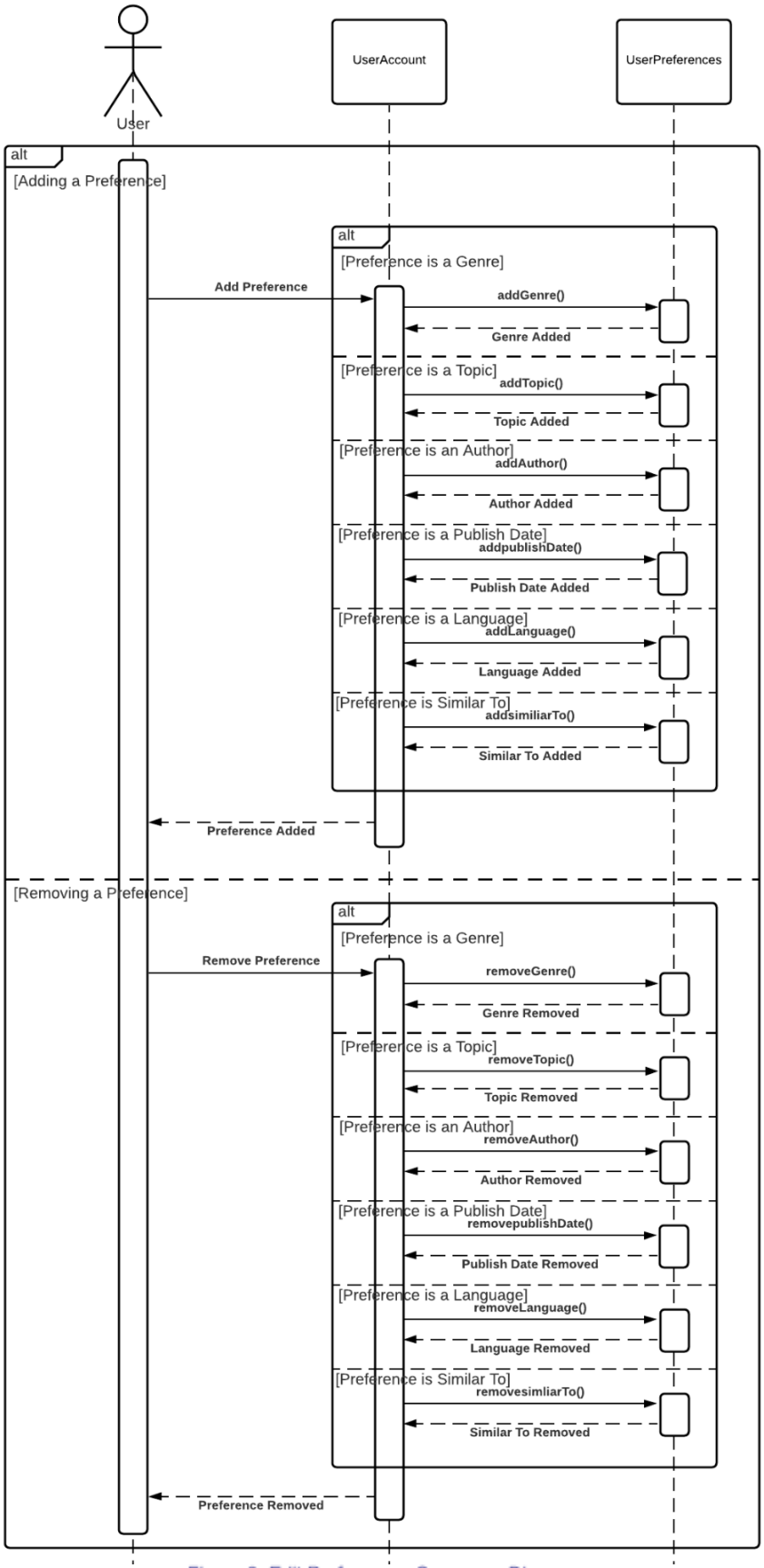

Figure 2. Edit Preferences Sequence Diagram

View Preferences

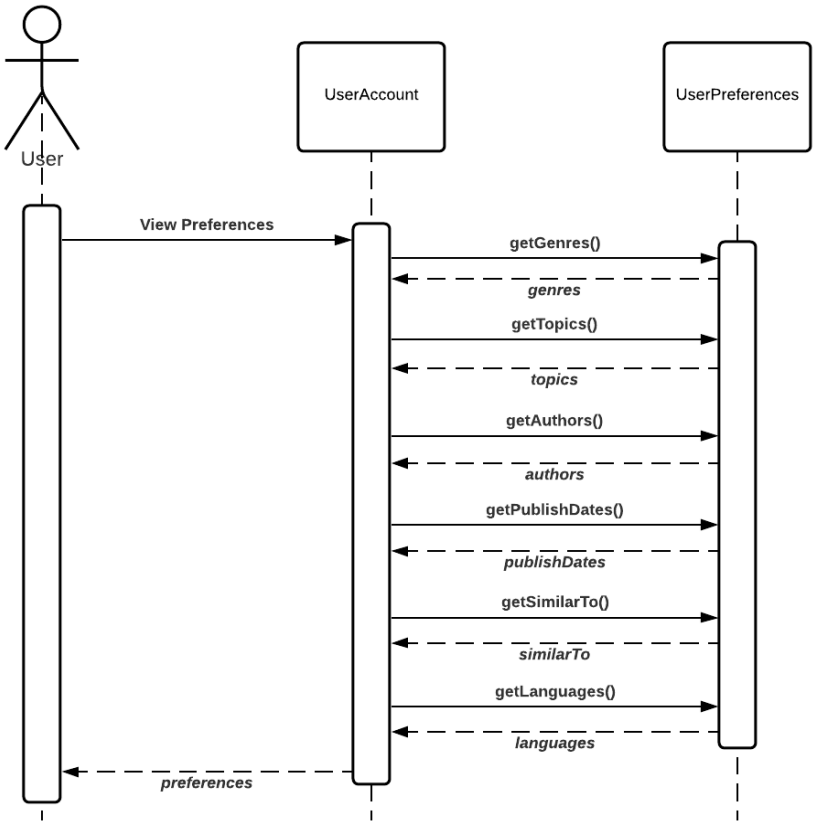

Figure 3. View Preferences Sequence Diagram

Edit Favorites List

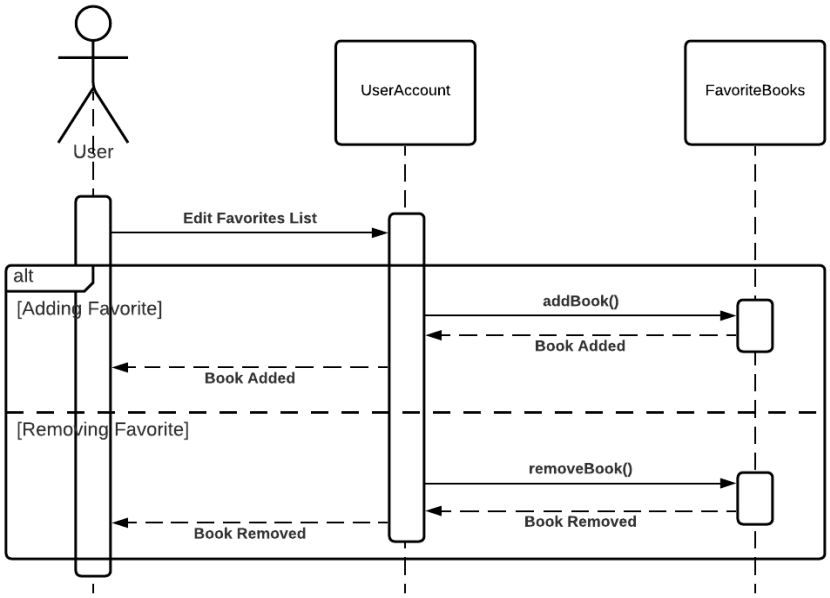

Figure 4. Edit Favorites List Sequence Diagram

View Favorites List

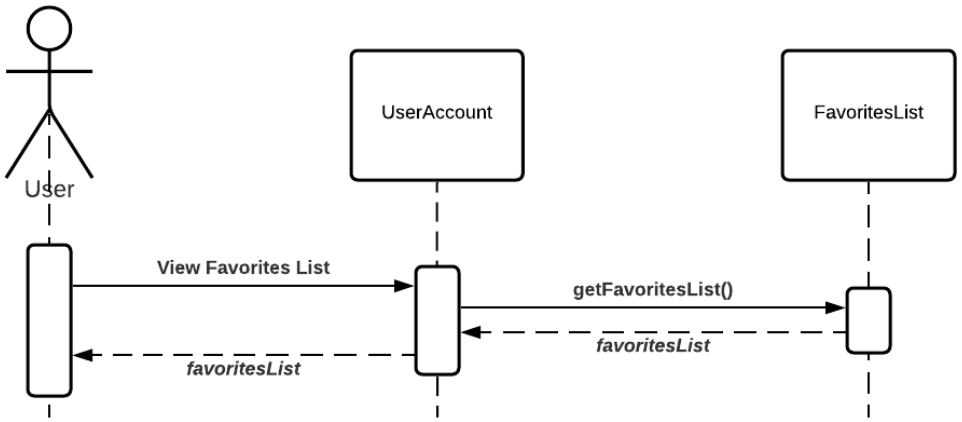

Figure 5. View Favorites List Sequence Diagram

Remove Recommended Book

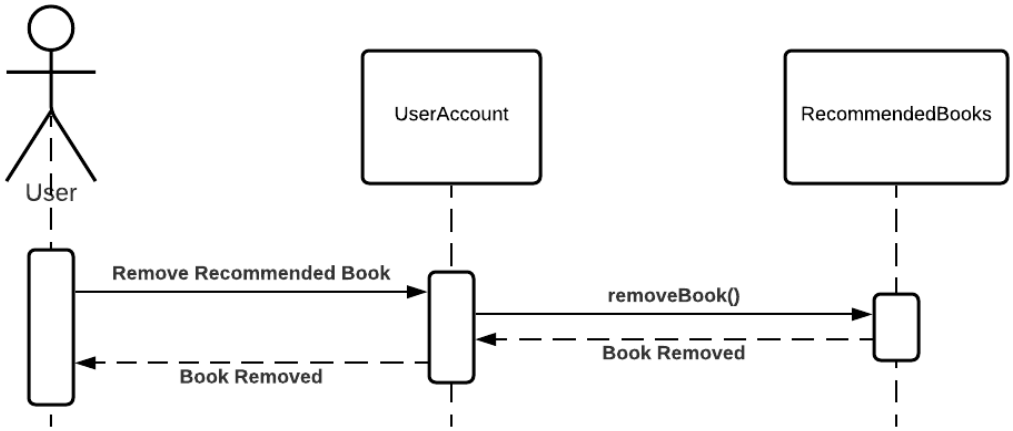

Figure 6. Remove Recommended Book Sequence Diagram

View Recommended Books

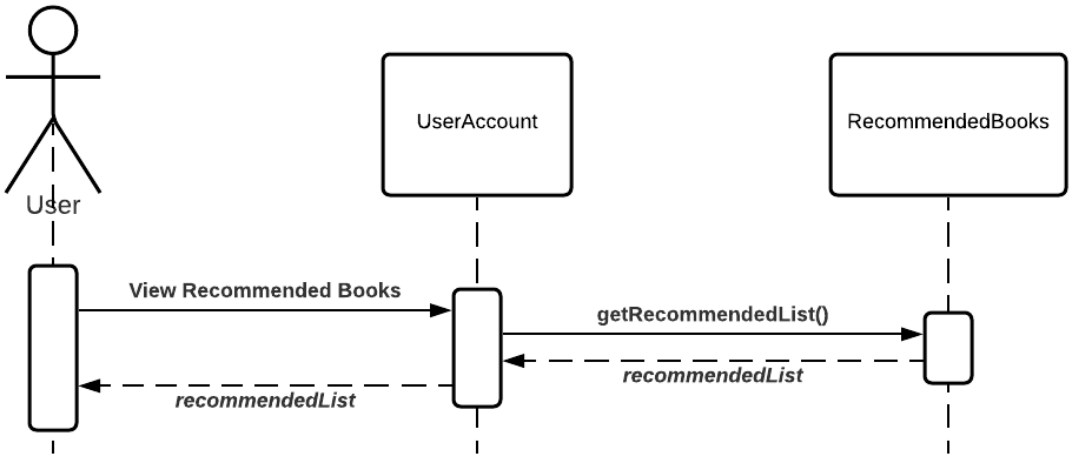

Figure 7. View Recommended Book Sequence Diagram

Search for Books

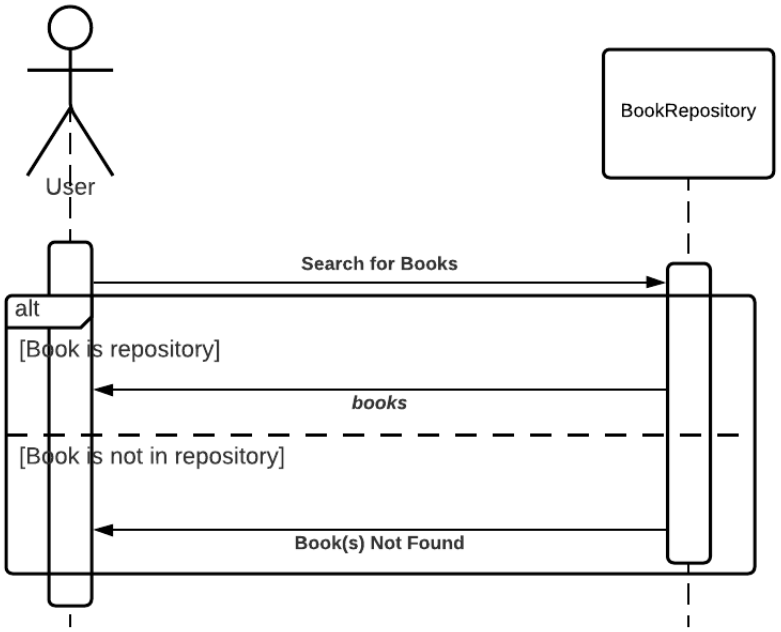

Figure 8. Search for Books Sequence Diagram

Class Diagram

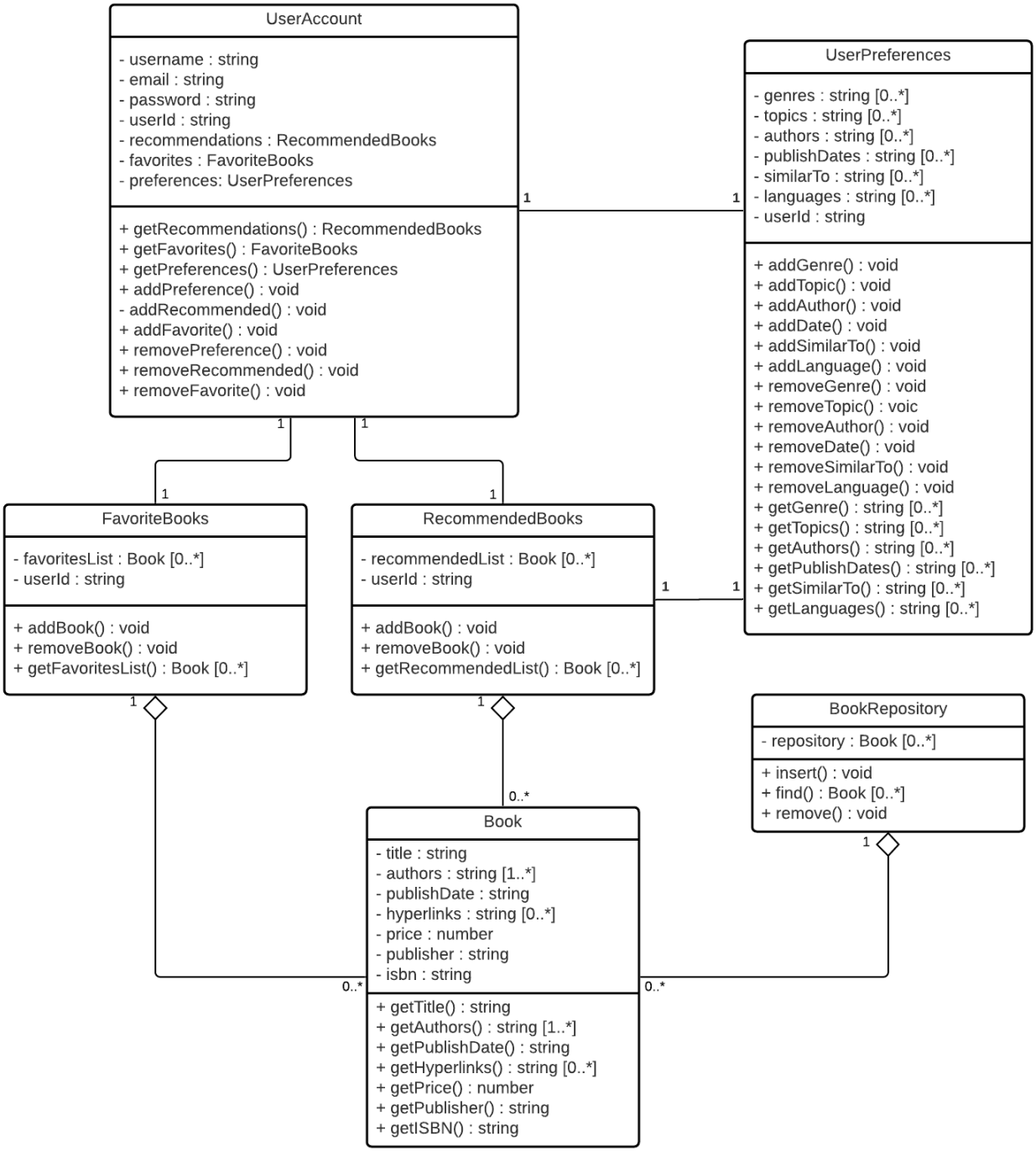

Figure 9. UML Class Diagram

Architectural Design

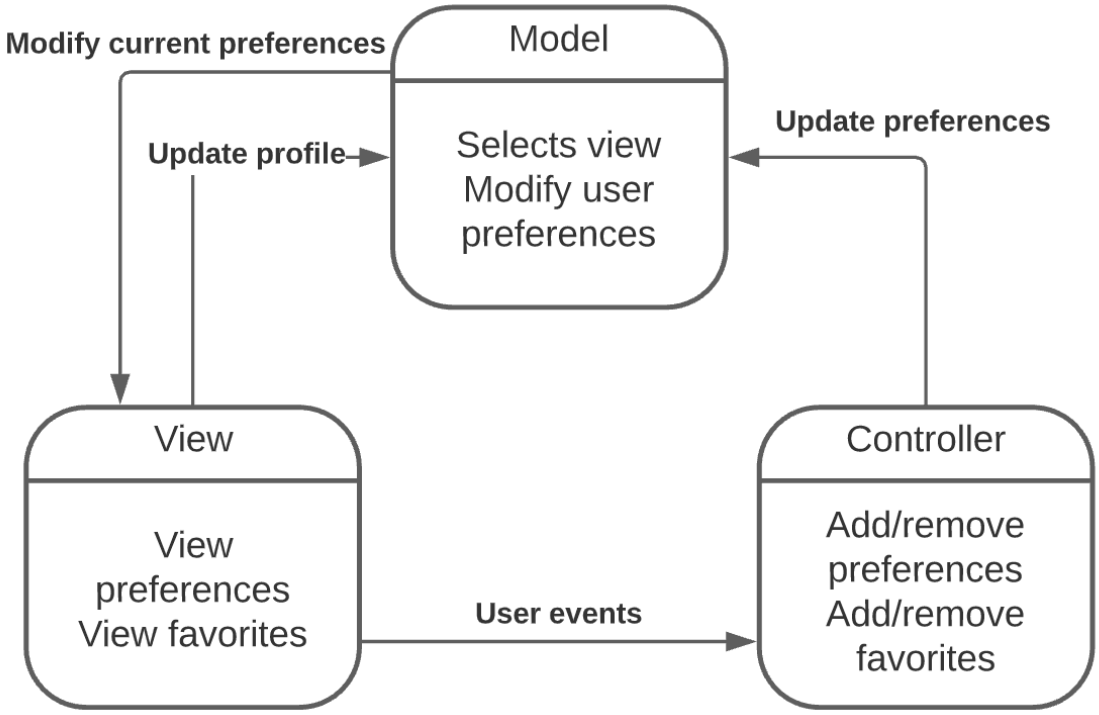

Figure 10. Architectural Design MVC Diagram

Project Deliverable 2 Content

Project Scheduling, Cost, Effort, Pricing, Duration, and Staffing Estimations

Project Scheduling

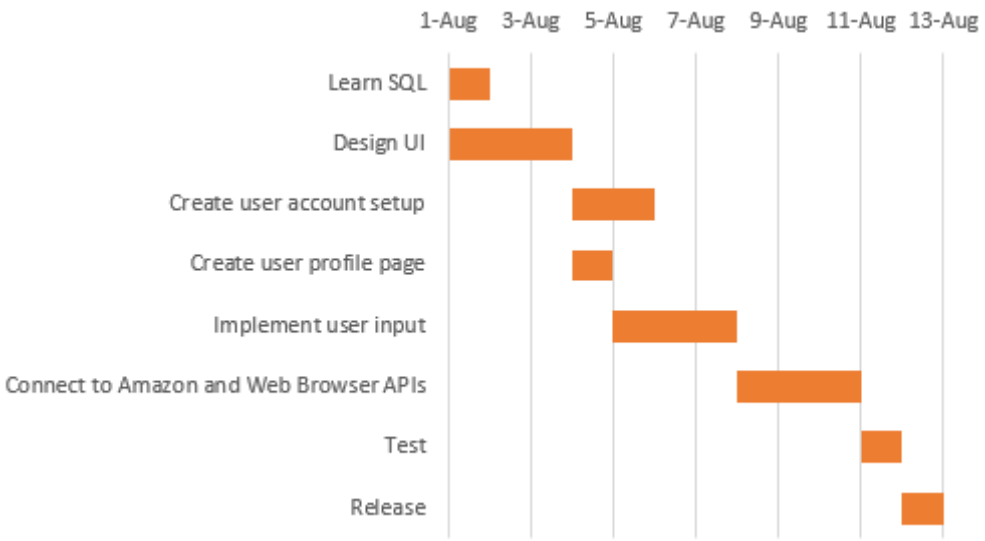

Figure 11. Project Schedule

This project is intended to take approximately two weeks to complete, starting on August 1st and ending on August 13th as shown in Figure 11. This schedule is assuming a typical 8-hour workday, from 9am to 5pm. Weekends are not intended to be included, but assuming all tasks do not go exactly according to this schedule, weekends can be accounted for. Furthermore, in our effort estimates, none of the complexities of our functions were not labeled as advanced. As such, in the event that problems do arise and the project is delayed, the lack of advanced complexity should allow for quick solving time.

Cost, Effort, and Pricing Estimates

For our project, the algorithmic estimation technique we chose to use was the Function Point technique. How we got the counts for each function category:

User inputs - (initial variable instatiations) username, email, password, user ID, favorites, preferences, genres, topics, authors, publish dates, similar books, languages (12)

User outputs - (what system is giving out) show favorites, show preferences, show recommendations, show book title, show book author, show publish date, show hyperlinks, show book price, show book publisher, show ISBN (10)

User queries - (any other functionalities users make) add favorites, add preferences, remove favorites, remove preferences, find books, change genres, change topics, change authors, change publish dates, change similar books, change languages (11)

Data files / relational tables - (total amount of tables and relational tables) user accounts, user preferences, list of favorite books, list of recommended books, book table, book repository database, user preferences to book repository database (7)

External interfaces - (external platforms connected to application) Amazon, web browser extension (2)

	Function Category	Count	Complexity			Count X Complexity
			Simple	Average	Complex	
1	Number of user input	12	**3**	4	6	36
2	Number of user output	10	**4**	5	7	40
3	Number of user queries	11	3	**4**	6	44
4	Number of data files and relational tables	7	7	**10**	15	70
5	Number of external interfaces	2	5	**7**	10	14

Table 1. Function Points

GFP = 36 + 40 + 44 + 70 + 14 = 204 FP

PC:

1. Does the system require reliable backup and recovery? 4
2. Are data communications required? 2
3. Are there distributed processing functions? 0
4. Is performance critical? 4
5. Will the system run in an existing, heavily utilized operation environment? 4
6. Does the system require online data entry? 3
7. Does the online data entry require the input transaction to be built over multiple screens or operations? 3
8. Are the master files updated online? 3
9. Are the inputs, outputs, files, or inquiries complex? 3
10. Is the internal processing complex? 3

11. Is the code designed to be reusable? 3
12. Are conversion and installation included in the design? 3
13. Is the system designed for multiple installations in different organizations? 3
14. Is the application designed to facilitate change and ease of use by the user? 5

Number	Complexity Weighting Factor	Value
1	Backup and Recovery	4
2	Data Communications	2
3	Processing Functions	0
4	Performance	4
5	Existing Operating System	4
6	Online Data Entry	3
7	Multiple Screen and Operation Input Transaction	3
8	Online Master Files	3
9	Complexity of Inputs, Outputs, Files, and Inquiries	3
10	Internal Processing	3
11	Reusable Code	3
12	Conversion and Installation	3
13	Multiple Different Organizational Installations	3
14	Ease of Use	5

Table 2. Complexities

PC = 4 + 2 + 0 + 4 + 4 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 5 = 43
PCA = 0.65 + 0.01(43) = 1.08
FP = GFP * PCA = 204 * 1.08 = 220.32 FP

Using the values computed from Table 1 and Table 2, we calculate the following (assuming productivity is 20 FP per person-week):

E = FP / productivity = 220.32 / 20 = 11.016 = about **12 person-weeks of effort estimation.**
D = E / team size = 12 / 7 = about 1.71 weeks = about **2 weeks duration estimation.**

Estimated Cost of Hardware

In order to develop and maintain our application, we need the proper equipment. At the core of the development of any application are the servers that support the application. The average expense for servers that support maintenance of apps in general is $20-$60 a month [1]. For our application, it is neither extremely simple, nor too complex, so we estimate the cost of servers to be around $37 monthly[1].

Overall, the estimated cost of hardware would be $444 annually since we will use a cloud-based database (MySQL), so we don't need to allocate money for physical database hardware.

Estimated Cost of Software

We decided to use MySQL for deploying the application on a cloud-based server. We chose the MySQL Enterprise Edition License, which provides us with features such as transparent data encryption, enterprise backups, firewall, and authentication. The cost of the MySQL Enterprise Edition License is $5000 annually [2].

To publish our application to the Google Play store, we must obtain a Google Play Developer Account, which is a one-time fee of $25 [3]. To publish our application to the iOS App Store, we need to join the Apple Developer Program, which costs $99 per year [4]. For our web domain, the cost is $15 per year to maintain [5].

We also will use WebStorm, a proprietary IDE for JavaScript to develop and test the application from client-side, and version control. The licensing cost of WebStorm is $129 per year [6]. During development, we also use Postman to develop and test our API, costing $288 per year [7].

Overall, the estimated cost of software per year is $5600. This includes costs of software used in development as well as licensing fees for publishing the app across various platforms.

Estimated Cost of Personnel

Based on calculations using the function point cost modeling technique shown in Table 1 and Table 2, our team concluded that approximately 7 programmers, the same size as our team, would be required to code the end product. We can also estimate training and the project collectively having a duration of about 2 weeks and assume that 20 FP per person-week translates to each programmer working roughly 20 hours per week.

Training costs could be minimal and conducted on an individual basis with accessible, detailed documentation; however, if additional training is needed, it costs about $35 per hour to hire a software consultant [8]. Let us assume if a consultant is hired, they consult for about 10 hours.

According to ZipRecruiter, a software engineer working at a startup makes an average of $54 per hour [9]. With all this information in mind, we estimate the cost of personnel to range around $15000-$15500 at the most.

Development Team: 7 programmers * $54 per hour * 20 hours per week * 2 weeks = $15120

Plus hiring a consultant: $15120 + ($35 * 10 hours) = $15470 total Personnel Cost.

Testing Plan

We decided to perform black-box testing on our units because even if we have access to the code itself, we want to make sure the external behavior of the unit meets specifications regardless of the internal implementation.

The unit we are testing is the subprogram of the UserAccount component that removes a given book from the user's favorites list. The requirements for this subprogram are as follows. The subprogram is called *removeFavorite* and accepts a Book object as its parameter. It then removes the book object from the user's favorites list. If the parameter passed in is an invalid data type or if the book does not exist in the user's favorites list, then no action is performed and the favorites list remains unmodified.

Based on those behavior requirements, we developed these test cases and expected results.

1. Parameter is a Book object that exists in the favorites list.
   Expected Result: Book object specified by the parameter is removed from favorites list.

2. Parameter is a Book object that does not exist in the favorites list.
   Expected Result: No modification is made to favorites list.

3. Parameter is invalid because it is a primitive data type.
   Expected Result: No modification is made to favorites list.

4. Parameter is invalid because it is an incorrect object type.
   Expected Result: No modification is made to favorites list.

Our unit passed all four test cases, as shown by the following JUnit test results in Figure 12 (p. 21).

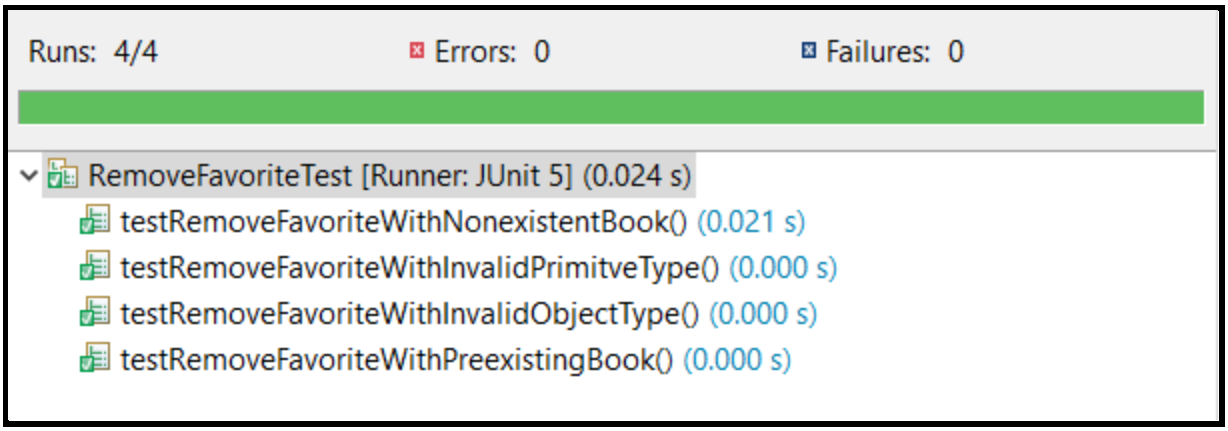

Figure 12. JUnit Test Results

The source code for the units and tests can be found on our project's GitHub repository.

Comparison of Work with Similar Designs

Within the book software industry, the two works that are most similar to our own project are Goodreads and Tailored Book Recommendations (TBR). To simply put it, they are both applications where users can get book reviews and recommendations based on their own book interests [10, 11].

Goodreads is an application where users can search upon a database of books to get information on any book. When a user is setting up their profile, they are asked multiple questions about books they have read, genres they like, and other things in order to create a simple recommendations list that the user can start off with. Along with this, Goodreads has a book database that users can use in order to get more information about books they are interested in and get information like reviews, ratings, and other recommendations from other users [10].

Our project, Bound 2 Be (B2B), compared to Goodreads is similar in the way that it is a book database system that involves users interacting with each other. Just like in Goodreads, Bound 2 Be has the ability for users to search up any book within its database to get information on the book as well as recommendations [10]. Goodreads also asks about the preferences of the user using a questionnaire, but unlike Bound 2 Be, it doesn't go very in depth. In Bound 2 Be's questionnaire, we will be asking more in depth questions revolving around 'themes and tropes' in order to get better recommendations for our users [10].

Tailored Book Recommendations (TBR) is another book service application, where you tell a bibliologist your preferences in a book, your goals with reading, what kinds of books that you would like to discover, and any other specific themes you are looking for in books. Once you are finished with updating your account with these specific preferences, Tailored Book Recommendations then prompts you to pay for a subscription where they personally give you 3 recommendations and a detailed description on why they chose those recommendations, and optionally send you the hardcover books as well [11].

Compared to our project, Bound 2 Be (B2B), Tailored Book Recommendations focuses more on the recommending side of book services. Their main goal is to recommend you books based on more open ended questions like “what kind of books you would like to discover and any specific themes within books” that suggest that real people actually shift through and find you recommendations personally, rather than an algorithm making matches with books that you favor with other books that other users have favorited [11]. While Tailored Book Recommendations has an actual bibliologist making recommendations for a rate, Bound 2 Be aims to get rid of this cost that Tailored Book Recommendations has, while still having a more in depth recommendation feature [11]. We call this our ‘dynamic recommendation algorithm’ where we not only consider your personal in-depth preferences questionnaire, but also recommendations that other users have to give.

Conclusion

Our group’s intention with Bound 2 Be was to create a smarter book-matching service. We wanted to allow users to input their likes, dislikes, preferences, etc., and our matching algorithm would provide a book recommendation for them that was catered to their interests. The hope is that finding books that people genuinely enjoy will rekindle their love of reading, as so many lose that passion as time goes on. This was our plan from the beginning and it did not deviate much from the initial idea as our project scope was of reasonable complexity. The cost estimations also seem reasonable upon examination, as with the tools available today to developers a simple concept such as Bound 2 Be can take shape within several weeks, or a few months at most, with a skilled development team. In conclusion, we believe the requirements, diagrams, estimations, testing plan, and overall system concept were well planned and thoroughly explained throughout this project.

References

[1] Michael Georgiou, “Cost of Mobile App Maintenance In 2021 and Why It’s Needed,” *Imaginovation*, 06-Feb-2020. [Online]. Available: https://www.imaginovation.net/blog/importance-mobile-app-maintenance-cost/. [Accessed: 10-Nov-2021].

[2] “MySQL Enterprise Edition,” *MySQL*. [Online]. Available: https://www.mysql.com/products/enterprise/. [Accessed: 07-Nov-2021].

[3] “Publishing your first app in the play store: What you need to know,” *Android Authority*, 18-Dec-2020. [Online]. Available: https://www.androidauthority.com/publishing-first-app-play-store-need-know-383572/. [Accessed: 07-Nov-2021].

[4] A. Inc., “Choosing a membership,” *Choosing a Membership - Support - Apple Developer*. [Online]. Available: https://developer.apple.com/support/compare-memberships/. [Accessed: 07-Nov-2021].

[5] Syed Balkhi, “How much does a domain name really cost? (expert answer),” *WPBeginner*, 26-Mar-2021. [Online]. Available: https://www.wpbeginner.com/beginners-guide/how-much-does-a-domain-name-really-cost-expert-answer/. [Accessed: 07-Nov-2021].

[6] “Buy webstorm: Pricing and licensing, Discounts - JetBrains Toolbox Subscription,” *JetBrains*. [Online]. Available: https://www.jetbrains.com/webstorm/buy/#commercial. [Accessed: 07-Nov-2021].

[7] “Postman api platform: Plans & pricing,” *Postman*. [Online]. Available: https://www.postman.com/pricing/. [Accessed: 07-Nov-2021].

[8] “Software Consultant Salary,” *ZipRecruiter*, 04-Nov-2021. [Online]. Available: https://www.ziprecruiter.com/Salaries/Software-Consultant-Salary. [Accessed: 11-Nov-2021].

[9] “Software Engineer Startup Salary,” *ZipRecruiter*, 04-Nov-2021. [Online]. Available: https://www.ziprecruiter.com/Salaries/Software-Engineer-Startup-Salary. [Accessed: 11-Nov-2021].

[10] *Goodreads.com*. [Online]. Available: https://www.goodreads.com/. [Accessed: 10-Nov-2021].

[11] *Mytbr.com*. [Online]. Available: https://mytbr.co/. [Accessed: 10-Nov-2021].