1Functional Requirements Specification  
Derive the use cases based on the requirements from Section 1 and Section 2 above.  
Note: If you wrote [user stories](http://en.wikipedia.org/wiki/User_story) instead of system requirements, you still need to write use cases. This fact is implicitly recognized by the introduction of “[epics](https://www.mountaingoatsoftware.com/blog/stories-epics-and-themes)”. If you like epics, write epics. (You know the old sayings, “Po-tay-to, to-mah-to,” or “Much of a muchness.”)

* 1. Stakeholders (Avani, Kutay)  
     Identify anyone and everyone who has interest in this system (users, managers, sponsors, etc.). Stakeholders should be humans or human organizations.
  2. Actors and Goals (Avani, Kutay)  
     Identify the *roles* of people or devices that will directly interact with the system, their *types* (initiating vs. participating) and the *goals* of the initiating actors.
  3. Use Cases
     1. Casual Description(Shaz,Alan)  
        For **all** use cases that you can think of (based on your System Requirements), write a *brief* or *casual* text description. List explicitly the requirements that each use case responds to.
        1. Case 1(Find recommended books)
           1. User→ enters the website and begins to type into the search bar
           2. System← Begins to predict titles based on letters already typed. Begins to pull data from the database
           3. User→ Select the title that matches what they wish to select
           4. User→ Hit enter after selecting all the books that they wished to input
           5. System← Creates table that stores all the books that the user input
           6. System← Starts running the similar match algorithm, and then stores them in the same file with the books that were entered
           7. System← Sends data to the ranking algorithm and begins placing top 50 books into tiers
           8. User→ Views books in list order from S→ C
        2. Case 2 (Wager between friends who say That is Certain book is a better match to a certain book. So is book C or book B a better match for book A)
           1. User→ Enter the name of book A into the search bar
           2. System← begins to find the name of the book in the database for predictive text
           3. User→ Selects the book from the predictive text
           4. System← Creates table that stores all the books that the user input
           5. System← Starts running the similar match algorithm, and then stores them in the same file with the books that were entered
           6. System← Use Analysis Algorithm to rank the books form S→ C
           7. User→ Use search function to find book B and C and see which book is more closely related to book A
     2. Use Case Diagram(Shaz)  
        Draw the use case diagram with all the use cases. Indicate the relationships, such as <<include>> and <<extend>>.
     3. Traceability Matrix (Seymour)  
        Show how your system requirements map to your use cases. Calculate the priority weights of your use cases. The use cases with the highest priority should be elaborated and planned for the [first demo](https://www.ece.rutgers.edu/~marsic/Teaching/SE/demo1.html).

Traceability Matrix (For Enumerated Functional Requirements)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requirements | Priority Weight (of Requirement) | UC-1 | UC-2 | UC-3 | UC-4 |
| REQ-1 | 2 | ✓ | ✓ |  | ✓ |
| REQ-2 | 5 |  | ✓ |  |  |
| REQ-3 | 4 |  | ✓ |  | ✓ |
| REQ-4 | 4 | ✓ | ✓ |  |  |
| REQ-5 | 5 |  | ✓ | ✓ | ✓ |
| REQ-6 | 5 |  | ✓ | ✓ | ✓ |
| REQ-7 | 1 |  | ✓ | ✓ |  |
| REQ-8 | 4 | ✓ |  |  |  |
| REQ-9 | 2 | ✓ |  |  |  |
| REQ-10 | 4 |  |  | ✓ | ✓ |
| REQ-11 | 4 |  |  | ✓ | ✓ |
| REQ-12 | 2 |  |  |  | ✓ |
| REQ-13 | 4 |  | ✓ | ✓ | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Max Priority Weight (of Use Case) | 3 | 4 | 5 | 5 |
| Total Priority Weight (of Use Case) |  |  |  |  |

Use cases 3 and 4 are most definitely what makes our software unique compared to similar software available today, hence why they are given the highest weights. The specific detail of the use case 3 so essential to our idea is that it involves ranking the found books into tiers, and then comparing all the tiers to all the books who were entered. This specific tool is what our software does, that no other current software considers, hence why it is one of the crucial and innovative pieces of our project. Use case 4 is also one of the most important pieces of the software because it carries out the data from use case 3 to the user, and essentially displays to the user what our algorithms and findings would most recommend. Although use case 4 is typical with recommendation programs, it will function very specific and closely to how user case 3 will ‘tell’ it operate and display. Use case 1 is the same type of implementation that can be found in practically all web searches, hence why it is important but not as specific to our design; and case 2 does what most recommendation software does when a user inputs single book to find recommendations for. Use cases 3 and 4 is what really sets our idea apart from what’s currently available.

* + 1. Fully-Dressed Description(Shaz, Anthony, Akshat)  
       Select a **few most important** use cases and provide *detailed* (“fully dressed”) description. The “most important” use cases are indicated by your traceability matrix.

The two most important use cases, as shown from out traceability matrix, are use cases 3 and 4. In a nutshell, use case three takes all the data from use cases 1 and 2, then essentially creates the basis for which our program will conclusively decides what will be recommended to the user. Use case 4 plays the role of communicating the results to the user and also giving a brief plot description of each of the options while also indicating which of the tiers each book falls under. In order to describe use case 3 in detail, we must at first look at briefly describe what use cases 1 and 2 do. Use cases 1 and 2 for the most part store the user inputted books, and also creates a database for the books that are found through searching. These found books will then become the basis for which the program will rank (in terms of compatibility for the user based off of what the user likes)  
Your event flows must show step-by-step *every action that the initiating actor (“user”) can take* while running the given use case.

Use Case 3: Search Prediction

Primary Actor: User

Goal: To narrow down the search results to the same type of book

Participating Actors: A computer/mobile device

Preconditions: Software has all popular search data and the necessary algorithm

Postconditions: The user has received a list of books with similar titles

Flow of Events:

1. User→ Enters Harry Potter into the search bar.
2. System← begins to find Harry Potter in the database for predictive text
3. User→ Selects Harry Potter from the predictive text
4. System← Creates table that stores all the books that the user input
5. System← All types of Harry Potter books regardless of the specific title are displayed on the user interface
6. System← Use Analysis Algorithm to rank the books form S→ C
7. User→ Use search function to find book B and C and see which book is more closely related to book A

Use Case 4: Evaluate Recommendations

Primary Actor: User

Goal: To find the desired book from the recommendations

Participating Actors: A computer/mobile device

Preconditions: A list of books with the similar titles are retrieved from the database and ready to be used

Postconditions: The user has found the desired book from the predicted searches

Flow of Events:

1. User→Wants *Harry Potter and the Sorcerer’s Stone*
2. System← Performs algorithm and displays *Harry Potter and the Chamber of Secrets, Harry Potter and the Sorcerer’s Stone, Harry Potter and the Goblet of Fire*. The system takes the key word Harry Potter and finds any book that starts with those keywords. Everything that appears after those keywords are irrelevant
3. User→ Selects *Harry Potter and the Sorcerer’s Stone* since it is listed.
4. System← Opens the book for the user to read.
   1. System Sequence Diagrams   
      Draw the system sequence diagrams for the **few most important** use cases selected above - (Use Cases 3 and 4)



User Interface Specification (Vedanta)  
(Note: If your system prints some forms or generates periodic reports, this is also considered part of the user interface and the format of forms/reports must be specified in this section.)  
The user interface should be specified only for the use cases elaborated in the previous section (“fully dressed” use cases).

* 1. Preliminary Design  
     For a given use case, show step-by-step how the user enters information and how the results appear on the screen.  
     Use *screen* [*mock-ups*](http://en.wikipedia.org/wiki/Mockup) and describe exactly what fields the user enters and buttons the user presses. Describe *navigational paths* that the user will follow.   
     In case you are developing a graphics-heavy application, such as a video game, this is one of the most important sections of your report.
  2. User Effort Estimation  
     Select several typical usage scenarios and, as you walk through the flow of events, count and report the number of mouse clicks and/or keystrokes that are needed to accomplish the task. What fraction of these goes to user-interface navigation vs. clerical data entry?  
     See example in Appendix A: [User Effort Estimation](https://www.ece.rutgers.edu/~marsic/Teaching/SE/report1-appA.html).

Meeting Minutes: 90 minutes