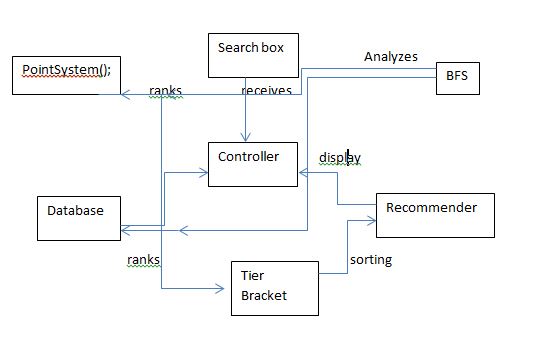
1. **Domain Analysis**

**Domain Model (Seymour and Anthony)**

**Draw the diagram.**

****

**Show the process of deriving the domain model**

In simplest terms, the domain model ‘s purpose is to take all the characteristics and operations of the requirements and use cases, then proceed to describe and illustrate how they interact with each other to make the program operate. The domain model allows for an in depth look at these interactions and gives a clearer look as to what is happening in the “backend” portions of the program. For our particular program, the main components of the program exist in the user interface, database, the manipulation of the data in the database, and then finally the end result outputted to the user. In essence, besides the input partition of the program, every piece of this project interacts with each other and depends on each output from the previous step. The user interface allows the user to input the books they’d like to search, and our particular search box will then interact with our current database of existing books in order to try and predict to the user the entire title of the book. The database in itself will be the host of where all the inputted books will reside (provided to the database from the user interface), where all the current books in the database resides, and where the inputted books along with the books for recommendations. So the database interacts with the user interface (search box), and also essentially plays a role with the data that will conclusively be communicated to the user. The manipulation of the data in the database will use the information gathered in the database, and begin to breakdown the significance of certain books in order to build possible recommendations to the user. So it will interact with the database’s given outputs, and proceed to filter out what will/will not be communicated to the user. The output will interact with the most updated version of the database (what’s left after all the manipulation), and output it chronologically to the user. So this will have heavy interactions will practically all aspects of the program except for the initial search box.

Provide text description of:

* + 1. Concept definitions (Anthony)

Controller - Coordinate actions of concepts associated with this use case and delegate the work to other concepts

Search box - receives the user's input as strings and displays a dropdown of search predictions for the user to potentially click on

Database - prepare database query that best matches the user’s search criteria and retrieve the records and retrieve the records from the database

Recommender - goes through tags with the highest multiplier

BreadthFirstSearch(); - looks for tags with the highest point value and traverses to other tags with the lowest points

PointSystem(); - gives tags points depending on how well they match the user’s reading preferences and search patterns

TierBracket(); - ranks books according to their points and puts them into their appropriate tiers for easy and fast recommendations

* + 1. Association definitions (Anthony)

Controller & Database - Controller passes the search requests to the Database

Controller & PointSystem() - Controller retrieves the data of points ready to be used.

BFS & TierBracket() - BFS looks for books with specific point values for the TierBracket() to sort them

TierBracket() & Recommender - Recommender suggests books to the user after looking up books that are associated with tags of the highest points

* + 1. Attribute definitions (Seymour)

Searchbox

Attributes: Text Input, Predetermine Title/Author

- Text Input: takes in inputs from the user in the form of text from a hardware keyboard (and potentially other forms of input depending on what we decide will be our select operating systems of compatibility)

- Predetermine Title/Author: Takes whatever text is in the search box and tries to predetermine what the possible title of the book might be, along with the books author. It will update as more and more characters are inputted into the search box

Database

Attributes: book database, input storage

-Book Database: This is where all of the books in the entire database will be stored (should already include the books that the user inputted in)

- Input Storage: This is separate database, where all of the inputs will be stored, and will be update and convert into a database containing the inputs and the books similar to the inputs

Recommender

Attributes: tag multiplier

-Tag Multiplier: will go through each of the similar books to the inputs (from the input storage) and quantify the tags of each of those books, with a multiplier, in order to associate each of the books with an overall number

PointSystem

Attributes: assigns points

-Assigns Points: takes each book from the input storage, and takes the information from the recommender and assigns a point quantity to each book based off of the accumulated tag multipiers

TierBracket

Attributes: Hierarchy

-Hierarchy: will takes the points for each book and rank each book from most points to least amount of points. It will then create cutoffs for each book, based off of the amount of points given to them, and break off the books into their respective tier

* + 1. Traceability matrix — show how your use cases map to your domain concepts. (Seymour)

Traceability Matrix of the Domain Concepts Along With The Use Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Domain Concepts | UC-1 | UC-2 | UC-3 | UC-4 |
| Controller | ✓ | ✓ | ✓ | ✓ |
| Search Box | ✓ | ✓ |  |  |
| Database |  | ✓ | ✓ |  |
| Recommender |  | ✓ | ✓ | ✓ |
| BreadthFirstSearch |  | ✓ | ✓ |  |
| Point System |  | ✓ | ✓ | ✓ |
| Tier Bracket |  | ✓ | ✓ | ✓ |
|  | Priority Weight Of UC-1: 3/5 | Priority Weight Of UC-2: 4/5 | Priority Weight Of UC-3: 5/5 | Priority Weight Of UC-4: 5/5 |

* 1. System Operation Contracts (Akshat, Alan)  
     Should be provided only for the operations of the fully-dressed use cases elaborated in Section 3.c), for their system operations identified in Section 3.d).

UC-3 Search Prediction

Preconditions:

* User had entered the books that they have read.
* Database is updated with the latest books
* Search algorithm is in working condition

Postconditions:

* User’s books’ suggestion is cleared from the search box for next input
* User has received a list of books that are recommended

UC-4 Evaluate Recommendations

Preconditions:

* The list of books recommended from UC-3 Search Prediction has been retrieved

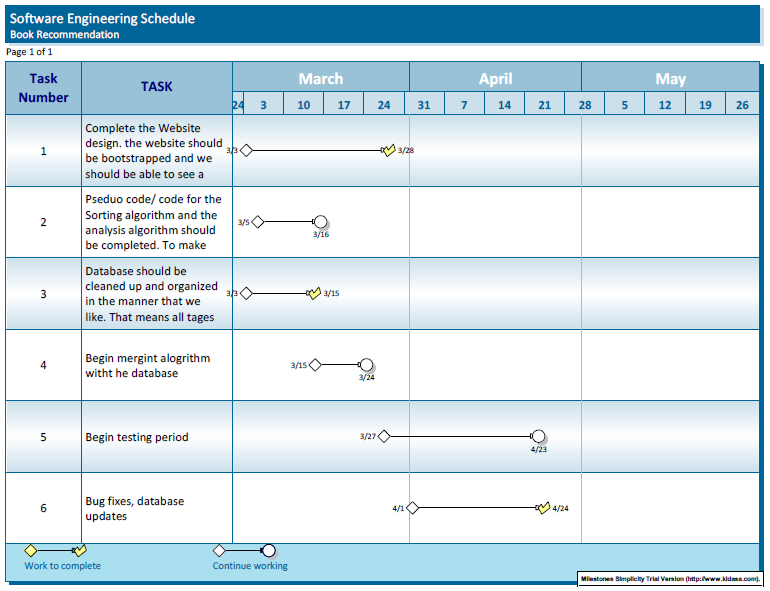
Postconditions:

* Books displayed can be sourced to read or buy online
* Books are sorted into the correct brackets based on the mathematical model
  1. Mathematical Model (Shaz, Alan)   
     Do you use any mathematical models? E.g., you may use a statistical model for stock price prediction, or a geometric model for computing the trajectories for animate figures in a video game.   
     If YES, describe precisely your model.
     1. Our mathematical modeling is when we use percentile brackets to break the recommendation into tiers. The way we do this is basically use the Bell Curve model. The reason we use this is because we want to make sure that there are only a few books at the S tier ranking. If we have only a selected few in the S rank, that makes the legitimacy of that tier better. That will be about the top 5% of books. The way that we are getting the point break up is by saving the scores of each book in out list. We take the one book that has the most points, set that as our max and then break that into each group. So for example let’s say that the top recommendation had a total point of 2500 points. S rank would be 2975-2500, A rank 2250-2974, B rank 2000-2249, C rank 1750-1999. That is how we are decided which books get placed in which tier

1. **Project size estimation based on** [**use case points**](https://www.ece.rutgers.edu/~marsic/Teaching/SE/projects-grading.html#QUANT)**. (Vedanta)  
   For Report #1, estimate the size of your project in terms of use case points *only for the use cases elaborated in this report*.***UC-1:*  
   External Inputs: User entering books, as well as selecting books to use for algorithm.  
   External Output: None.  
   External Inquiries: Dropdown with book names (on file) for user to select as inputs.  
   Internal Logical Files: List of books available to use for algorithm.  
   External Interface Files: None.

*UC-2:*  
External Inputs: User entering books, as well as selecting books to use for algorithm.  
External Output: Results of algorithm with books ranked based on similarity using tags.  
External Inquiries: Dropdown with book names (on file) for user to select as inputs.  
Internal Logical Files: List of books available to use for algorithm.  
External Interface Files: None.  
  
*UC-3:*  
External Inputs: User entering books, as well as selecting books to use for algorithm.  
External Output: Results of algorithm with books ranked based on similarity using tags.  
External Inquiries: Dropdown with book names (on file) for user to select as inputs.  
Internal Logical Files: List of books available to use for algorithm.  
External Interface Files: None.  
  
*UC-4:*  
External Inputs: None.  
External Output: Results of algorithm with books ranked based on similarity.  
External Inquiries: None.  
Internal Logical Files: None.  
External Interface Files: None.  
  
*Overview:*  
Although each use case (except the last one) requires inputs and results in an output, every use case overlaps every other use case, reducing the size of the overall project substantially. The final requirements will be: the algorithm (points allocation), analysis (statistical representation of results), user interface, and database (local) all of which can be applied to each of the use cases. Once we are able to receive inputs and display the accurate results, each use case should be available for testing, and the focus becomes optimization and increasing accuracy, as well as adding extra features given excess time.

1. Plan of Work (Vedanta, Shaz)  
   Describe what your group is planning to do after submitting report#1 until the end of the semester. Show the roadmap with projected milestones and dates by which you plan to accomplish them. Of course, your plans for the short term (next few weeks) should be much more detailed than further in the future.  
   Preferably, you should use [Gantt charts](http://www.ganttchart.com/) for planning and scheduling your project (also check: [Gantt chart basics: What it is, benefits, & alternatives](https://asana.com/resources/gantt-chart-basics)). (Rutgers students can download *Microsoft Project* at the [University Software Portal](https://software.rutgers.edu/).) Consider also creating your [product roadmap](https://blog.asana.com/2018/08/product-roadmap-tips-templates/).  
   Include the product ownership description from your [project proposal](https://www.ece.rutgers.edu/~marsic/Teaching/SE/proposal.html), and provide the breakdown of responsibilities: what each team member did so far, is currently doing, will do in the future, including management and coordination activities.

****

**Product Ownership Description**

**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

**Responsibilities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Did** | **Currently** | **Will Do** |
| **Shazidul  (Analysis Algo)** | Learned SQL and Python for compatibility | Working on Analysis Algo and sorting methods | Merge algo with database and site |
| **Vedanta (Sorting Algo)** | Created Github repository | Begin working on the algorithm | Finish algorithm and merge with database |
| **Avani**  **(Presentation and UI design)** | Documentation and Group Meeting Coordinator | Documentation and Group Meeting Coordinator | Documentation and Group Meeting Coordinator |
| **Anthony (Predictive Text)** | Contributed concepts for the functionality | Working with front end for site development | Will merg algo with site (he is the bridge between both) |
| **Alan (Database Entries)** | Database creator | Maintain and organizing database | Will maintain and update database on new entries |
| **Akshat** | Learned SQL and Python, gained understanding of Bootstrap and AWS | Currently working on front end site | Have predictive text up and merge with algo |
| **Seymour** | Research for point/tier system. | Working on sorting and selecting algo | Help merg the algo with the analysis algo and the site |
| **Kutay**  **(Presentation and UI design)** | Documentation | Documentation | Documentation and bug tester |

**Plan of Work:**

1. **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.

1. **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.

1. **References (EVERYONE)  
   The list of references should contain exact *references and URLs* of any material that is used in the project and doesn’t come from the textbook. If a reference is listed but not cited/mentioned in the main text, explain briefly in what way it was used**.