SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
SOFTWARE ENGINEERING

Design Report - Book Matching System

by

Mark Hatori
Taylor Mau
Mikhail Smelik

Santa Clara, California
October 13, 2017

Design Report - Book Matching System

Mark Hatori
Taylor Mau
Mikhail Smelik

Department of Computer Engineering
Santa Clara University
October 13, 2017

ABSTRACT

Currently there is a problem with matching students with books that would be interesting for them. The problem is that a professor does not have a way of presenting a list of books to students that would be easy to sort based on the preferences of the student. This causes the students to choose a book that might not match all their preferences. Another problem is that addition of books can go unnoticed by students and not factor into their choice of books. The solution is to create a web based system that will run on Santa Clara University?s Linux system in the Engineering Computing Center. The system will allow students to enter their preferences and be presented with a list of books that best match their preferences. Also the system will allow the professor to input in new books that will then be sorted by the system and presented to the students.

Table of Contents

List of Figures

List of Tables

1 Introduction

Students are more likely to be interested in reading if they are able to choose a book that seems fascinating to them. To help students find books that they are interested in, teachers at the elementary and middle school level put out books for their students to look at. However, this is a chaotic system where students may not see all the books or they might have to wait for other students to finish reading the description because the teacher has a limited number of books.

Students may struggle to find books that fit their preferred interests, especially for a large collections of books. Any additions to the collection by the professor may go unnoticed by the students, even if the new selections are a better match for a student.

Our solution will allow students to easily find books that they are interested in. We will be implementing a web based system that runs on Santa Clara University's Linux system in the Engineering Computing Center. A spreadsheet implementation of this would be easier to develop. However, we recognized that sometimes they can become disorganized and become hard to use. We will avoid this by creating our online system. We will be creating this for a teacher to use for one specific class.

Our solution offers a more efficient and effective way for students to select books to read. By implementing software that the students can all access at once, less class time can be devoted to the book selection process. Additionally, having software for this process allows for the teacher to have access to all the information from students? selections.

Teachers will be the ones who have the ability to add, edit and delete books in the database. Additionally, teachers will be able to add students to their class. Students on the other hand, are only able to search through the books by their preferred interests. Each party will have their own ability to login to their accounts. However, the student account will not have access to manipulate the database. When students search, they will be able to see a ranking of the books that match their preferences..

Our solution will be editable, the teacher can easily add, delete, or edit the description of books. Because the software could be implemented in an elementary or middle school setting, the User Interface should be easy to use. Ideally, our solution should also be self sufficient, in the sense that administrators do not need to set up the server constantly in order for the program to run. Also, since the goal is to have many students accessing the software at once, our solution should be able to support a class of students.

This web based system will allow a student to choose a book that is best matched to them based on their preferences out of a number of books offered by the teacher. We hope that this system is a more efficient and organized solution for the client.

2 Requirements

2.1 Functional

Critical

- Have a graphical user interface
- Allow the professor to edit books by adding, removing, and modifying preferences
- Professor adding students and removing them
- Allow students to reserve a book
- Allow the professor to view student selections
- Have a login system
- Priority order sort
- Teacher profile
- Student profile
- Editable preferences by students

Recommended

- Input a list of books
- Allow professor to modify the priority order
- Show which books are reserved

Suggested

- Allow the professor to sort books (test the sorting)
- Have separate logins pages for teacher and students

2.2 Non-Functional

Critical

- Encrypted password
- Scalable to many students
- Documented code
- Reliable to not crash Engineering Computing Center computers

Recommended

- Testability by client
- Menu Driven

Suggested

- Configurable

2.3 Design Constraints

- Web Based
- Run on Engineering Computing Center Linux machine

3 Use Cases

The use cases figure shows how the student client and teacher client can interact with the system.

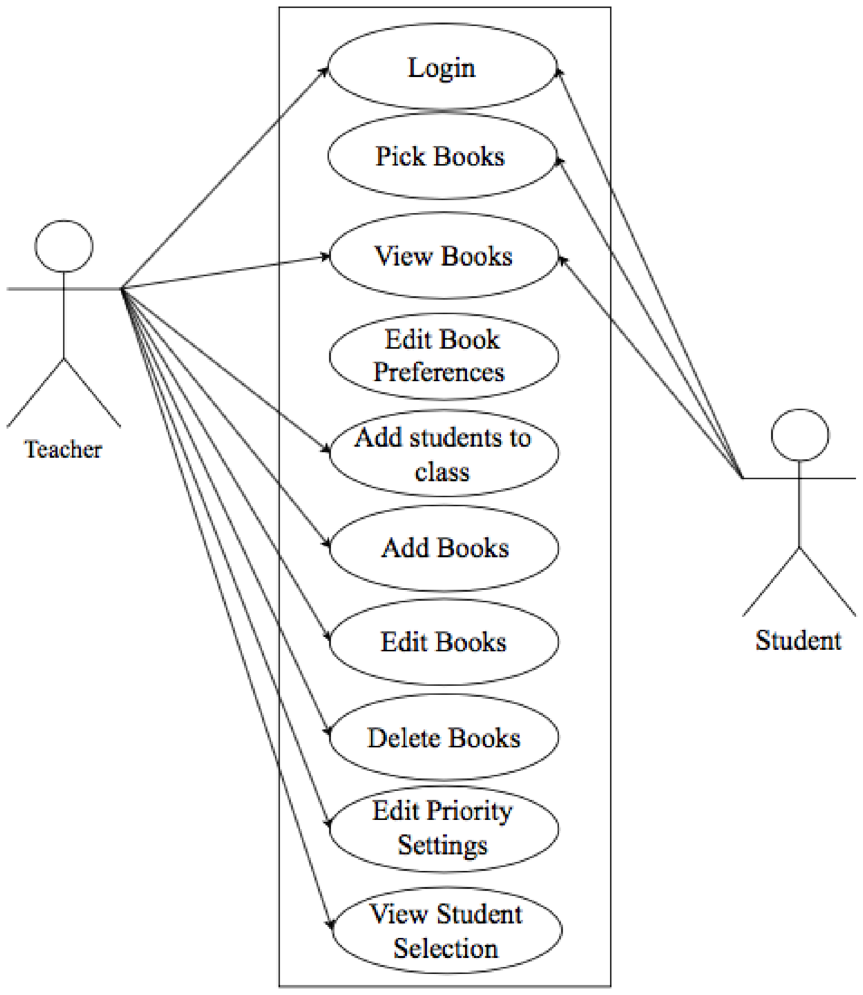


Figure 1: Use Cases

Login
Goal: Login to gain access to the website
Actor: Teacher Student
Preconditions: None
Steps: Enter credentials into dialogue boxes
Postconditions: Login credentials verified
Exceptions: None

Pick Books
Goal: Select the book
Actor: Student
Preconditions: Student must be logged in
Steps: Click the checkbox and click the submit button
Postconditions: Save changes
Exceptions: None

View Books
Goal: See the list of books
Actor: Student Teacher
Preconditions: Must be logged in
Steps: Click the show books button
Postconditions: None
Exceptions: None

Edit Book Preferences
Goal: Change search preferences
Actor: Student
Preconditions: Must be logged in
Steps: Click on the preferred option from dropdown menus and click update
Postconditions: Save changes
Exceptions: None

Add Student
Goal: Create a new student account
Actor: Teacher
Preconditions: Must be logged in, Student must not exist in database
Steps: click on the preferred option from dropdown menus and click update
Postconditions: Save changes
Exceptions: None

Add Book
Goal: Create new book in database
Actor: Teacher
Preconditions: Must be logged in, Book must not exist in database
Steps: Input information in a form and click Add
Postconditions: Save changes
Exceptions: None

Edit Book
Goal: Change Book information
Actor: Teacher
Preconditions: Must be logged in, Book must exist in database
Steps: Find section, change information, click Edit
Postconditions: Save changes
Exceptions: None

Delete Book
Goal: Remove book from database
Actor: Teacher
Preconditions: Must be logged in, Book must exist in database
Steps: Click checkbox to select book(s), click delete
Postconditions: Save changes
Exceptions: None

Edit Priority Settings
Goal: Change order, add, edit, or delete priority requirements
Actor: Teacher
Preconditions: Must be logged in
Steps: Click change order, change from dropdown menu, click Apply Changes.
Postconditions: Save changes
Exceptions: None

View Students? Selection
Goal: View a list of books selected by student
Actor: Teacher
Preconditions: must be logged in, Student must make selection
Steps: Click View students? Selection
Postconditions: None
Exceptions: None

4 Activity Diagram

The Student Activity Diagram shows the flow of actions a student can take when accessing the system.

4.1 Student Activity Diagram

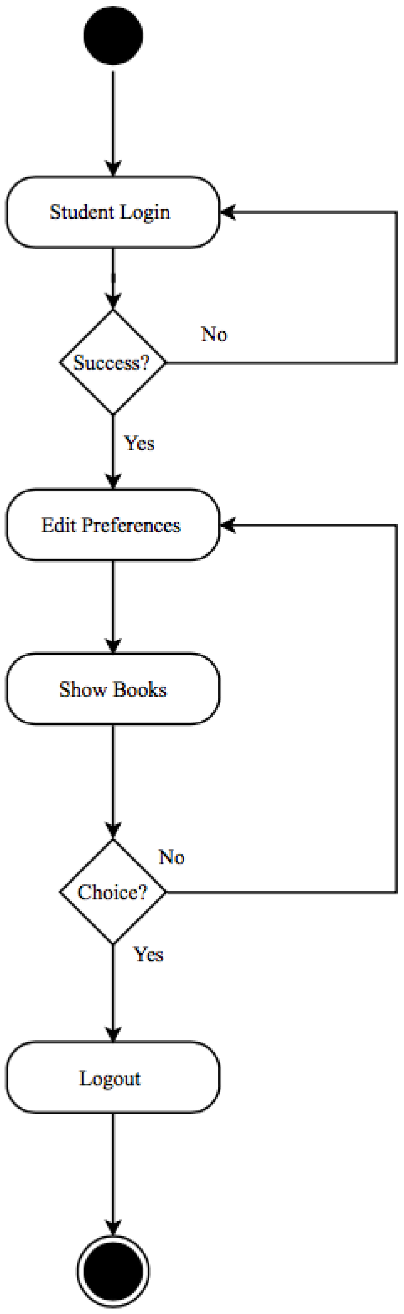


Figure 2: Student Activity Diagram

4.2 Teacher Activity Diagram

The Teacher Activity Diagram shows the flow of actions a teacher can take when accessing the system.

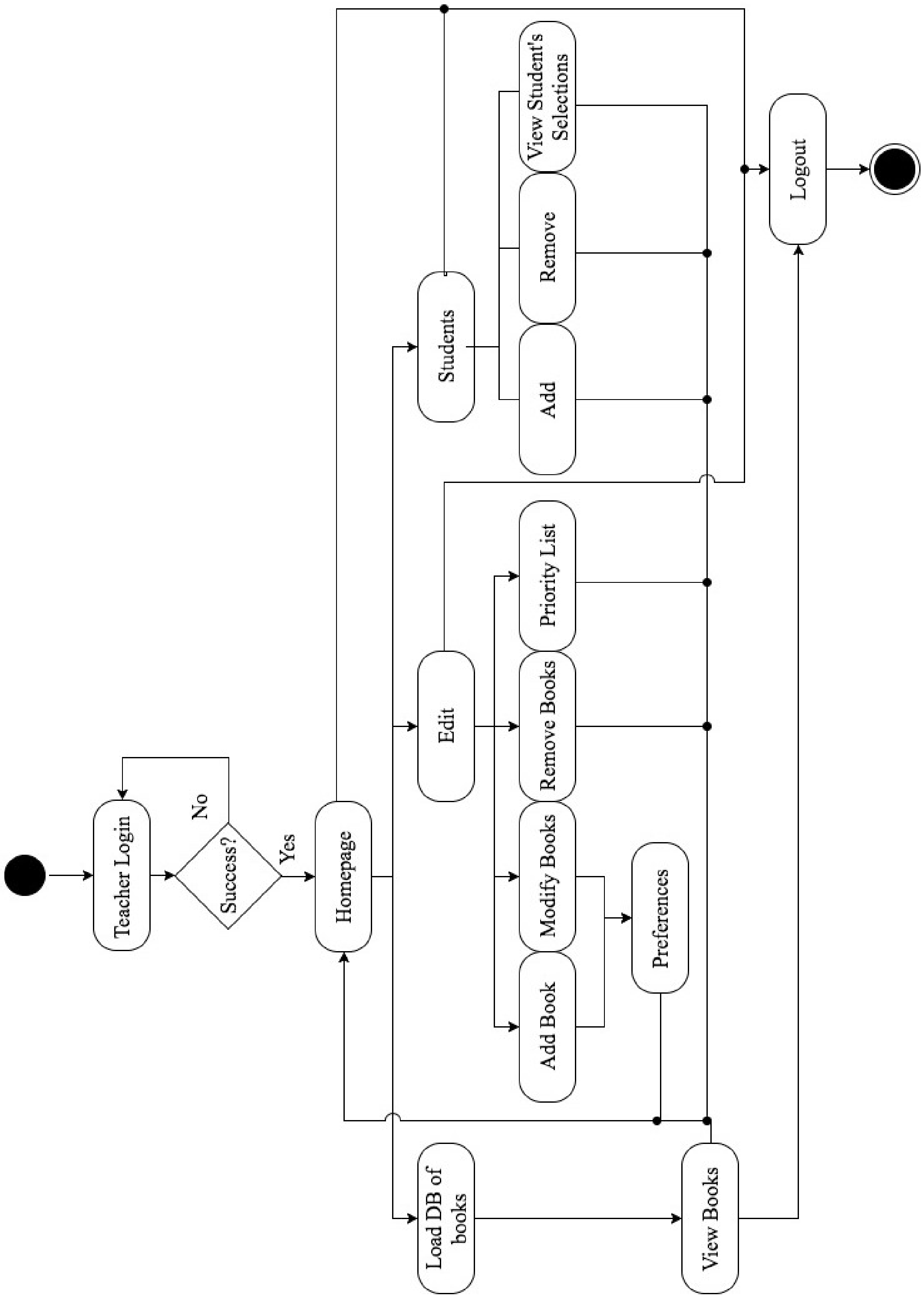


Figure 3: Teacher Activity Diagram

5 Conceptual Model

The Conceptual Model is a mockup of what the graphical user interface will look like. Everything shaded in a box will be a button.

This is the home page of the website that both students and teachers will see.

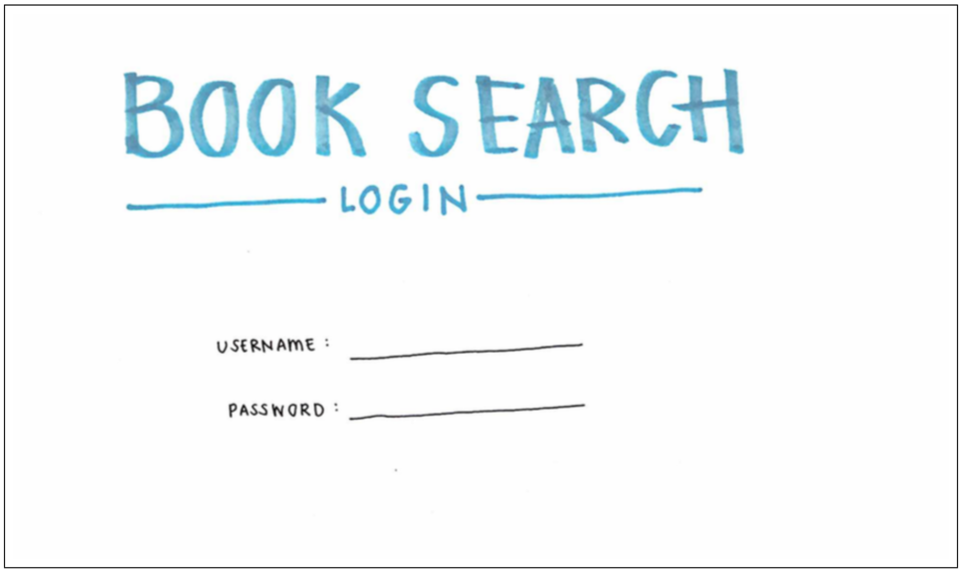


Figure 4: Home Page

The students will be prompted to in put their preferences.

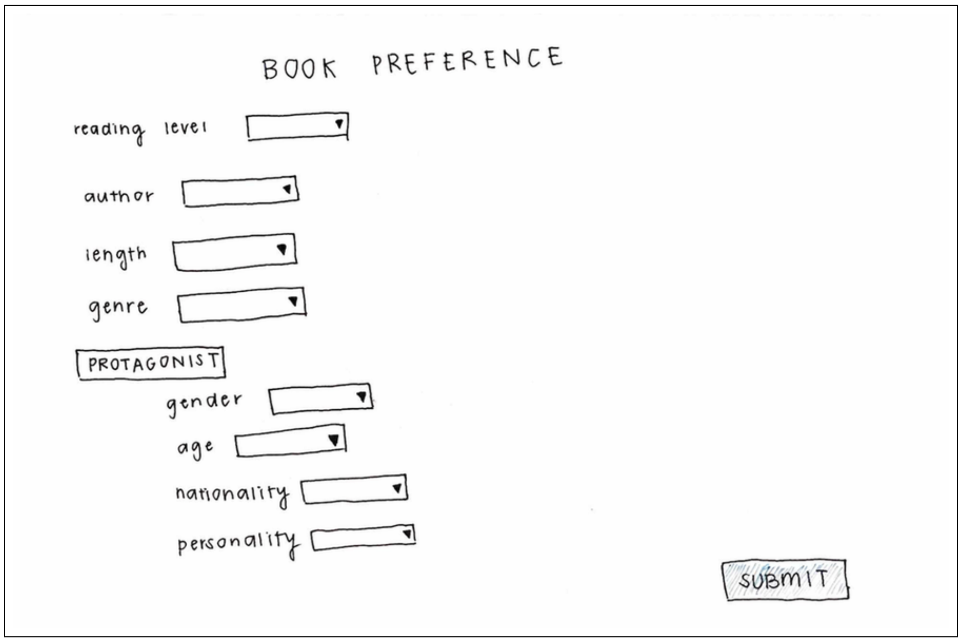
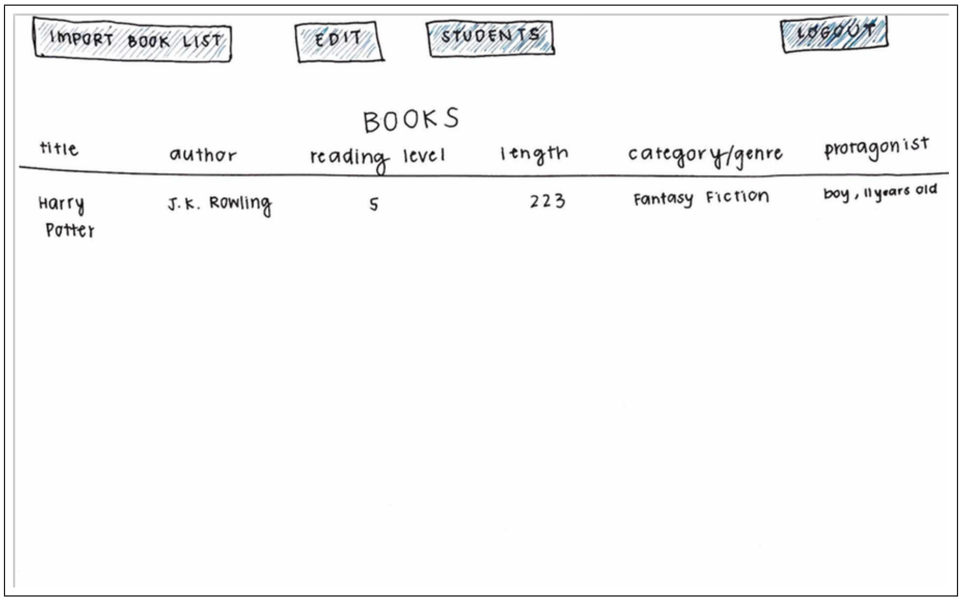


Figure 5: Book Preference Page

After teachers log in, they will see this page where they are then able to import book list, edit the list of books, and edit the students.



title	author	reading level	length	category/genre	protagonist
Harry Potter	J.K. Rowling	5	223	Fantasy Fiction	boy, 11years old

Figure 6: Teacher Home Page

After teachers are able to upload an excel spread sheet to add a bunch of books all at once.

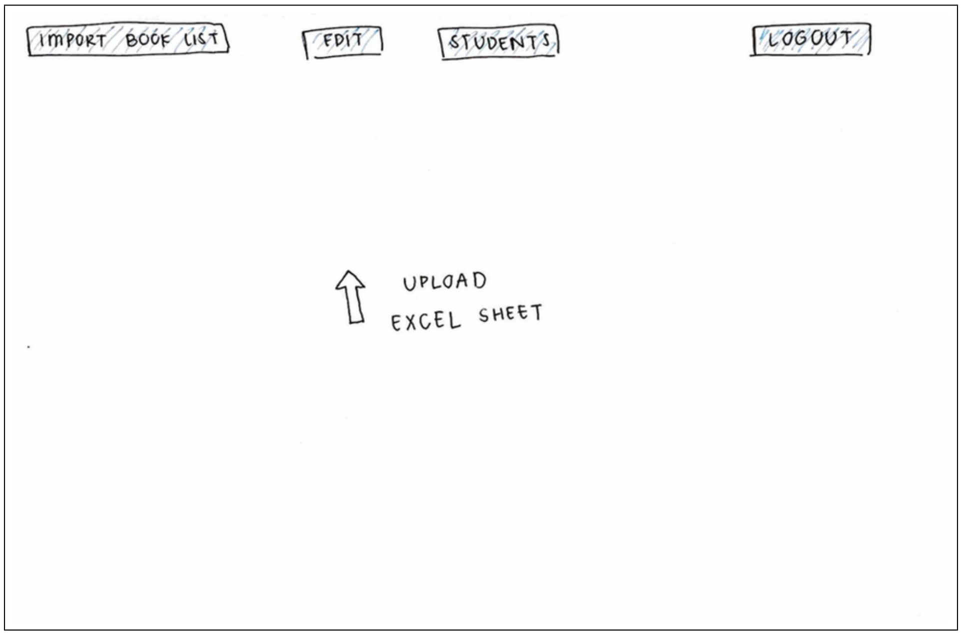
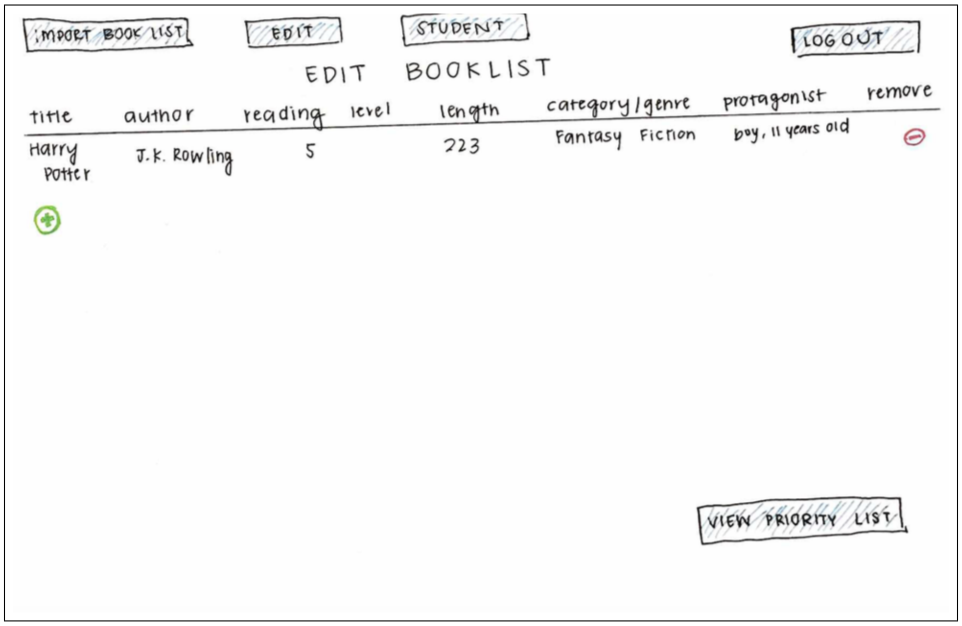


Figure 7: Import Book List Page

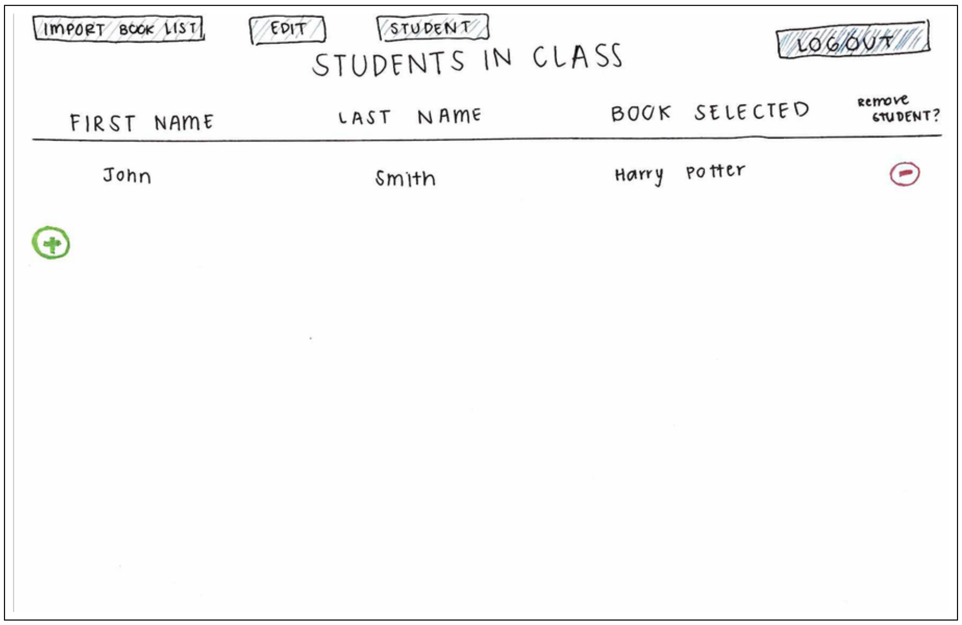
After teachers are also able edit the students in their class. The red button allows them to delete students while the green button on the bottom allows them to add students. The teacher is also able to view the priority list. To edit, the teacher is able to simply type on the page.



title	author	reading level	length	category/genre	protagonist	remove
Harry Potter	J.K. Rowling	5	223	Fantasy Fiction	boy, 11 years old	⊖
⊕						

Figure 8: Edit Book List Page

After teachers are also able edit the students in their class. The red button allows them to delete students while the green button on the bottom allows them to add students. The teacher is also able to view the priority list. To edit, the teacher is able to simply type on the page.



FIRST NAME	LAST NAME	BOOK SELECTED	remove STUDENT?
John	Smith	Harry Potter	⊖
⊕			

Figure 9: Edit Students Page

After teachers are able to edit the priority list which changes the order in which the books are sorted by.

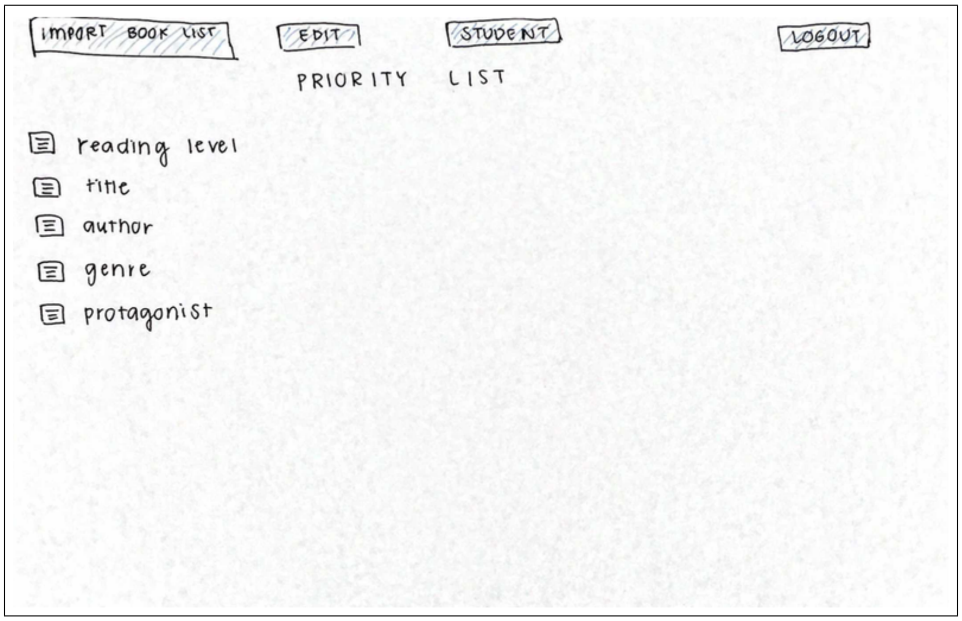


Figure 10: Priority List Page

6 Architectural Diagram

The architectural diagram shows the architecture that will be implemented for the solution

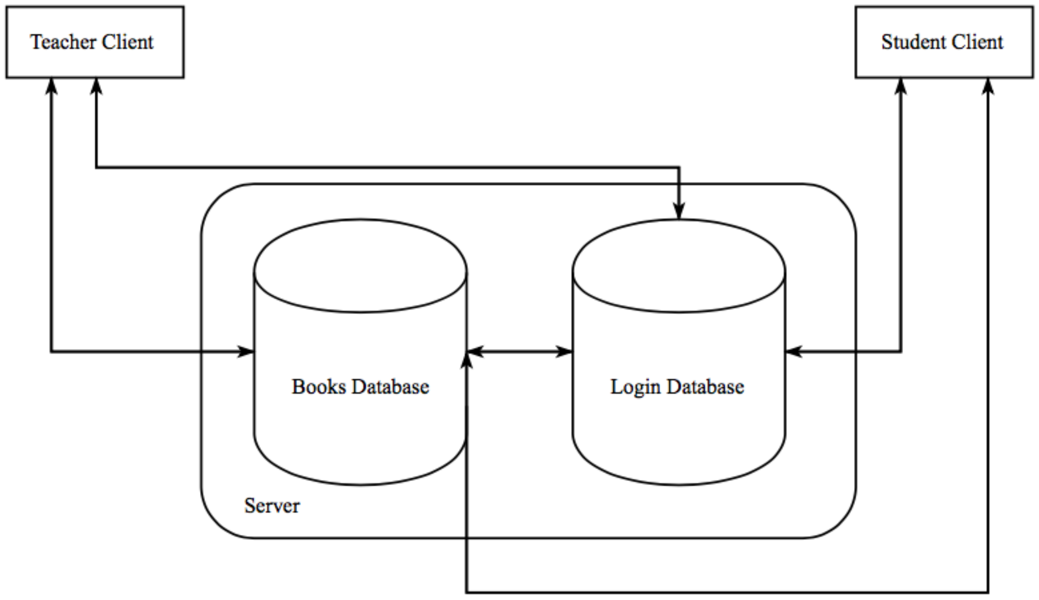


Figure 11: Architectural Diagram

7 Design Rationale

For the backend of our project, we plan on using the Oracle database software from Design Center using SQL as the language. We chose this software because it is available in the Engineering Center Computer Lab. Similarly, we chose to use SQL as our database language because it is available in the computer lab, and simple to use. We also plan on using ASP.NET to connect the database with our website because it is compatible with the Oracle Database. We will be using PL/SQL, the procedural language designed to work with SQL and Oracle databases.

For the frontend of our website, we plan on using HTML as it is standard for web browsers. We also plan on using CSS to provide the style for our website. We also would like to use Bootstrap to make the front end coding simpler and the website look more professional. We are also planning to use Javascript for transitions and to add the database to the website.

8 Test Plan

For the testing we will use black box, white box and user testing. We want to make sure that our system is working correctly from all sides, but especially from the side of the user. Our goal was to create a user friendly system so that students will be able to use this system without having to be explained how the system will work. We plan to work with students of a similar age in order to figure out how well the User Interface works. We will also do some black box testing. This is because we want to make sure that the system performs like how the user thinks it should. White box testing is important from our perspective because we want to be sure that the system is behaving properly. An example of what we would white box test is checking to see if the database updates the list of books correctly. We hope to do extensive testing to make sure the end user is happy with their system.

9 Risk Analysis

The risk analysis table shows the severity of the problems that can be experienced while creating the program. The impact of the risk was calculated by multiplying the probability of the risk happening by the severity of the risk.

Table 1: Risk Analysis Table

Risk	**Consequences**	**Probability**	**Severity**	**Impact**	**Mitigation Strategies**
Failure to link database to front end	System does not operate	0.5	9	4.5	Using database and front end software that is know to link together well
Failure to deploy solution to the Web	Client will not be able to use the system	0.5	8	4	Use software that reliably deploys programs to the web and deploy the solution ahead of time
Narrow range of operability	System becomes less useful to the client	0.4	5	2	Follow the design requirements to verify that all the requirements presented by the client are fulfilled
Difficult to navigate the system	Client will have a hard time using the system	0.3	6	1.8	Do extensive user testing to verify that the system is easy to use
Failure to authenticate login	Client cannot currently use the program	0.2	7	1.4	Extensive testing of the login system, confirming that it performs as expected
Crash Engineering Computing Center computer	System down	0.1	10	1	Extensive testing of the system to make sure that

10 Development Timeline

The development timeline shows the schedule, division of labor, and due dates for the project.

Timeline

- Oct. 10 Mockup Front End
- Oct. 11 design report due
- Oct. 13 Set up github
- Oct. 18 Design Review
- Oct. 23 Front End Done
- Nov. 8 Back End Done/ Deploy to website
- Nov. 8 - 15th User Testing and flow chart testing
- Nov. 15th Initial Operational System Demo / Final presentation
- Nov. 15 start final report
- Nov. 26 Finish Final report and then revise
- Nov. 29th Final Report Due/ Final operational demo

Design

- Oct 10 Front End Mockup
- Oct 23 Front End

Implementation

- Oct 13 GitHub
- Oct 18 Database
- Nov 8 Backend/Deploy

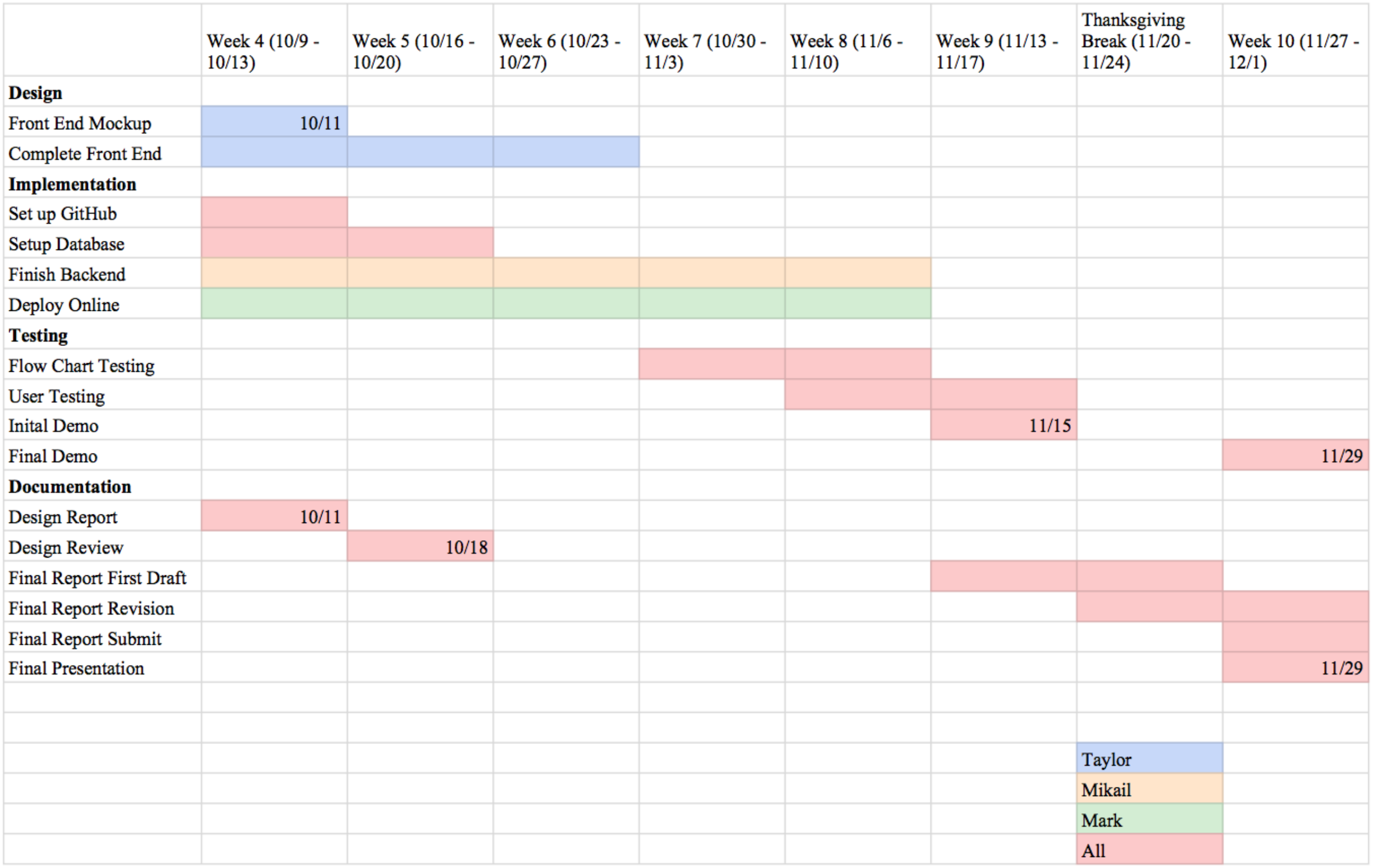
Testing

- Nov 5 - 7 Flow Chart Testing
- Nov 8 - 15 User Testing
- Nov 15 Initial Demo
- Nov 29 Final Demo

Documentation

- Oct 11 Design Report

- Oct 18 Design Review
- Nov 15-26 Final Report First Draft
- Nov 26 - 29 Final Report Revision
- Nov 29 Final Report Submission
- Final Presentation



	Week 4 (10/9 - 10/13)	Week 5 (10/16 - 10/20)	Week 6 (10/23 - 10/27)	Week 7 (10/30 - 11/3)	Week 8 (11/6 - 11/10)	Week 9 (11/13 - 11/17)	Thanksgiving Break (11/20 - 11/24)	Week 10 (11/27 - 12/1)
Design								
Front End Mockup	🟦 10/11							
Complete Front End	🟦	🟦	🟦					
Implementation								
Set up GitHub	🟥							
Setup Database	🟥	🟥						
Finish Backend	🟧	🟧	🟧	🟧	🟧			
Deploy Online	🟩	🟩	🟩	🟩	🟩			
Testing								
Flow Chart Testing				🟥	🟥			
User Testing					🟥	🟥		
Inital Demo						🟥 11/15		
Final Demo								🟥 11/29
Documentation								
Design Report	🟥 10/11							
Design Review		🟥 10/18						
Final Report First Draft						🟥	🟥	
Final Report Revision							🟥	🟥
Final Report Submit								🟥
Final Presentation								🟥 11/29
							🟦 Taylor	
							🟧 Mikail	
							🟩 Mark	
							🟥 All	

Figure 12: Development Timeline