SpartaSave

Team QuietCoders

CS 160 - 04: Software Engineering

Dora Do, Jack Kang, Kevin Tan, Loveleen Kaur

Table of Contents

Functional Specification

Problem Statement

Students are overspending on required textbooks for college courses. Students should be made aware of the many online options to buy textbooks to have the most stress-free, practical and cost-efficient college experience possible.

Objective

Help students have one resource to buy textbooks, allowing them to find the cheapest option based on various listings on online sites (i.e. Amazon). Students may also buy/sell amongst each other for easy, local transactions similar to the way Craigslist is setup.

Functional Requirements

There are six functional requirements we would like our system to be able to do and/or what our users should be able to do:

1. Users shall be able to post a textbook to sell, and remove their listing once sold.
2. Users shall be able to search for a textbook using ISBN, author, and/or title.
3. Users shall be able to add their own courses.
4. The system shall be able to retrieve prices of a given textbook.
5. The system shall be able to compare prices through student postings and online retailers such as Amazon.
6. The system shall be able to organize the prices of textbooks from low to high and based on their condition, new or used.

Nonfunctional Requirements

There are also four non-functional requirements we have listed that will address areas of usability, performance and supportability:

1. The system must support multiple browsers.
2. The homepage must be able to load in a few seconds.
3. The interface must be straight-forward to use and user friendly.
4. The system must display the results in an easy-to-read format.

Use Cases

Our user will be a San Jose State student. Figure 1. displays the use case diagram with six use cases. For example, as a SpartaSave user, they will be able to search for a textbook, post a textbook to sell and add a running list of their courses. We will further explore these three use cases thoroughly with the use case descriptions below.

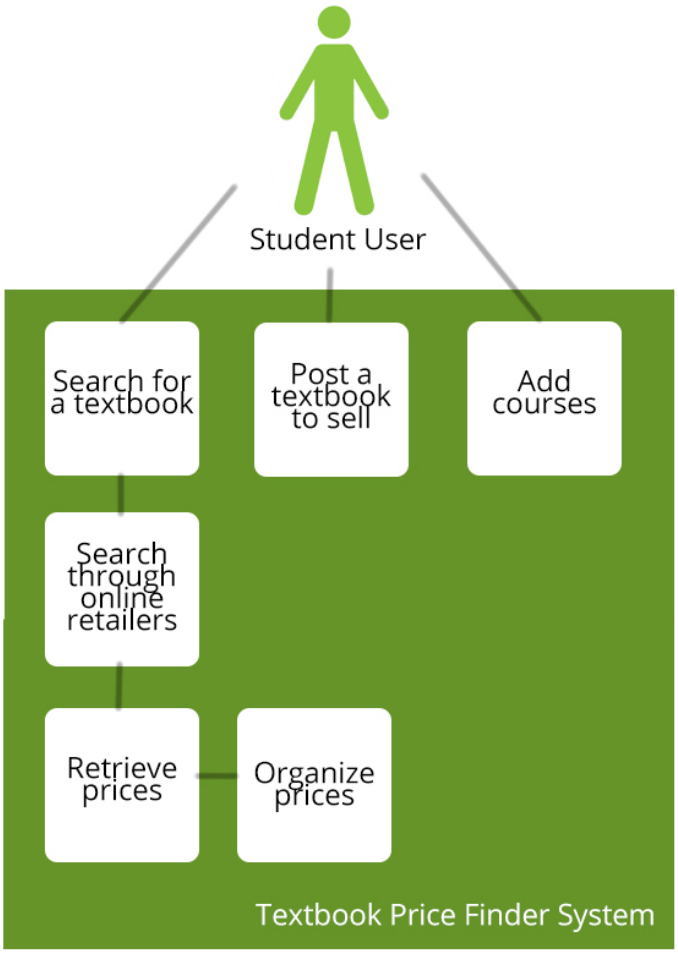


Figure 1. Use Case Diagram

Use Case Description

Use Case:	Student user adds courses
Product Name:	SpartaSave
Team:	Quiet Coders
Date:	9/15/14

1. Goal

Users shall be able to add their own courses.

2. Summary

Users will sign in to their account and add courses and their information. The site will save and show any updates of the user's course list.

3. Actors

- Student User

4. Preconditions

- User is on the home page

5. Trigger

User inputting courses

6. Primary Sequence

Step	Action
1	User enters username and password
2	User selects "add courses"
3	User inputs course information for each class

7. Primary Postconditions

- After step 1: System responds with prompt menu that includes:
  1) Add courses
  2) Look up a book
  3) Sell a book

4) Sign out

- After step 2: System prompts use to enter course information
- After step 3: System displays user's updated course list

8. Nonfunctional Requirements

- The interface must be straight-forward to use.
- The system must display the results in an easy-to-read format.

Use Case Name:	Student posts a textbook to sell
Product name:	SpartaSave
Team:	Quiet Coders
Date:	9/15/14

1. Goal

Users shall be able to post a textbook to sell on the site.

2. Summary

User will enter all the information applicable for the book. The system will then add the book to the listings along with the other books for sale.

3. Actors

- Student User

4. Preconditions

- User has logged into their account

5. Trigger

- User clicks on "Sell Textbook"

6. Primary Sequence

Step	Action
1	User selects "Sell Textbook"
2	System displays form to sell a book
3	User enters "Discrete Mathematics with Applications" for title field, "Susanna Epp" for author field, "9780495391326" for ISBN field, "25" for price field, and selects "Like New" for condition field
4	System displays "Successfully posted textbook" and displays a back option

7. Primary Postconditions

- User's textbook is posted on the listings
- Textbook is added on their account

8. Nonfunctional Requirements

- The interface must be straight-forward to use.
- The system must display the results in an easy-to-read format.

Use Case Name:	User searches for a textbook
Product Name:	SpartaSave
Team:	Quiet Coders
Date:	9/15/14

1. Goal

Users shall be able to search for a textbook using ISBN, author, and/or title.

2. Summary

User will search for a textbook they wish to purchase. The system will return all the matching results.

3. Actors

- Student User

4. Preconditions

- User is on the homepage or any page that has the search bar

5. Trigger

- User clicks on the search button

6. Primary Sequence

Step	Action
1	User types "Java Servlets and JPS" in the search for a textbook bar.
2	System displays all books matching that title and their corresponding prices and book condition, new or used

7. Primary Postconditions

- Matching results are displayed with their corresponding prices

8. Alternative Sequences

Alternative Postconditions

- System cannot find any matching results. System displays message, “could not find the textbook you were looking for. Would you like to use ISBN?”

9. Nonfunctional Requirements

- The interface must be straight-forward to use.
- The system must display the results in an easy-to-read format.

Conceptual Design

Based on the functional requirements and use cases, we came up with four major features we wanted to implement for our users and a UI design that was quite modern in appeal and responsive in display.

Major Features

1. A search engine that allows students to find textbooks from various sites (i.e. Amazon, Chegg, etc.)
    a. Sorts by price, new or used
    b. Searches user postings as well
    c. List other editions of the same textbook and their prices

2. Users are able to sell their own textbooks and/or buy them from other users
    a. Create posting by category to sell books and remove the listing once it is sold
    b. Browse posted books by category
3. Users can keep a running list of courses they have taken
4. Internal messaging system
    a. Buyers and sellers can arrange details of the transaction

Major Modules

Figure 2 displays the major modules that will be playing a role in our system; the client, the server, the data sources and the data integration server.

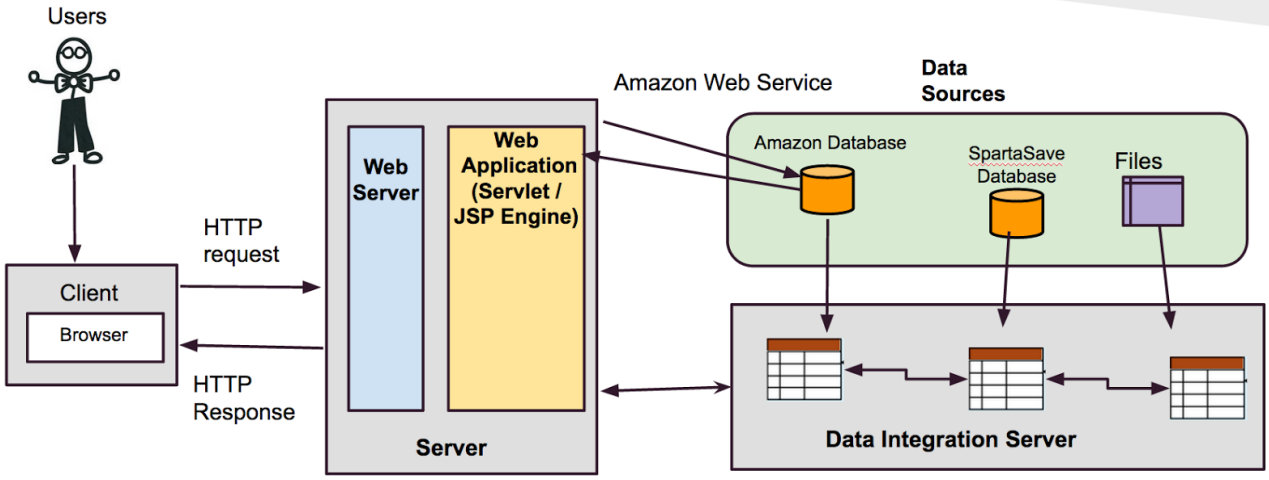


Figure 2. High Level Architecture Diagram of our Major Modules

The user interacts with the client web browser on their computer to enter books they want to search for or to sell via HTML forms on the web application. The data on the HTML form is sent to servlets on the web application for processing to determine the next action. To search for a book, the servlet will need to access the SpartaSave database, as well as use the Amazon Web Service (SOAP / REST request) to access the Amazon database. A servlet will then take the resulting responses and integrate the data from our different sources and store them on our combined database. Then a servlet can return the HTTP response back to the client browser as a dynamically generated HTML page / JSP with the search results. Or to post a book, the user will enter data on the HTML forms describing the book to sell. The servlet will take that data and store it on the SpartaSave database for search. If the user is just browsing the student postings, the servlet will return a JSP with books organized by category.

Design Mock-up

Once we set our goals and understood the architect of our system, we started to design the user's view. We started with a few wireframe mockups on Balsamiq.com. Below is a mockup design of the use case for a user searching for a textbook using our search engine.

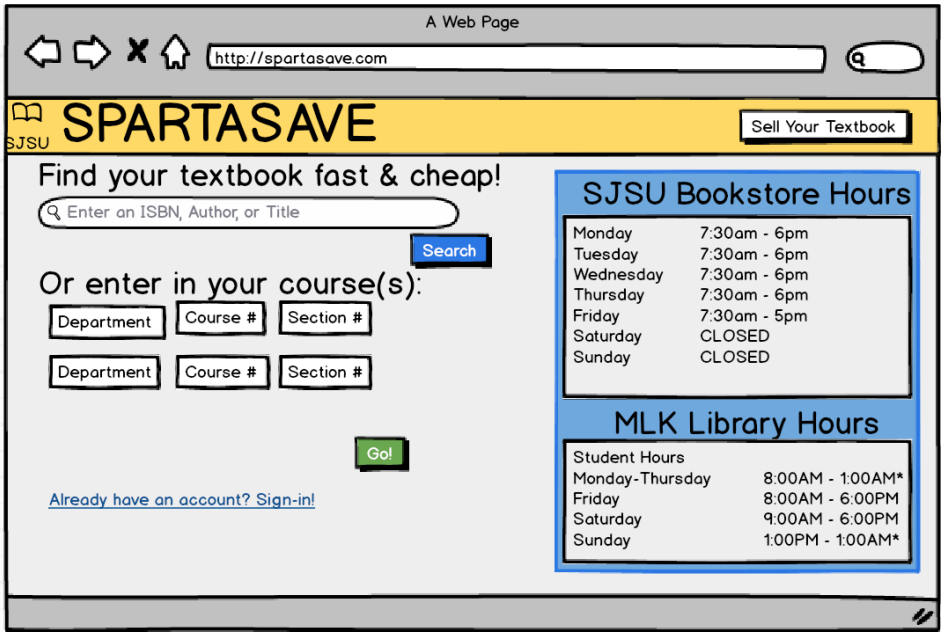


Figure 3. [index.jsp] User will enter in the title, author or ISBN of the textbook in the search bar and hit "Search"

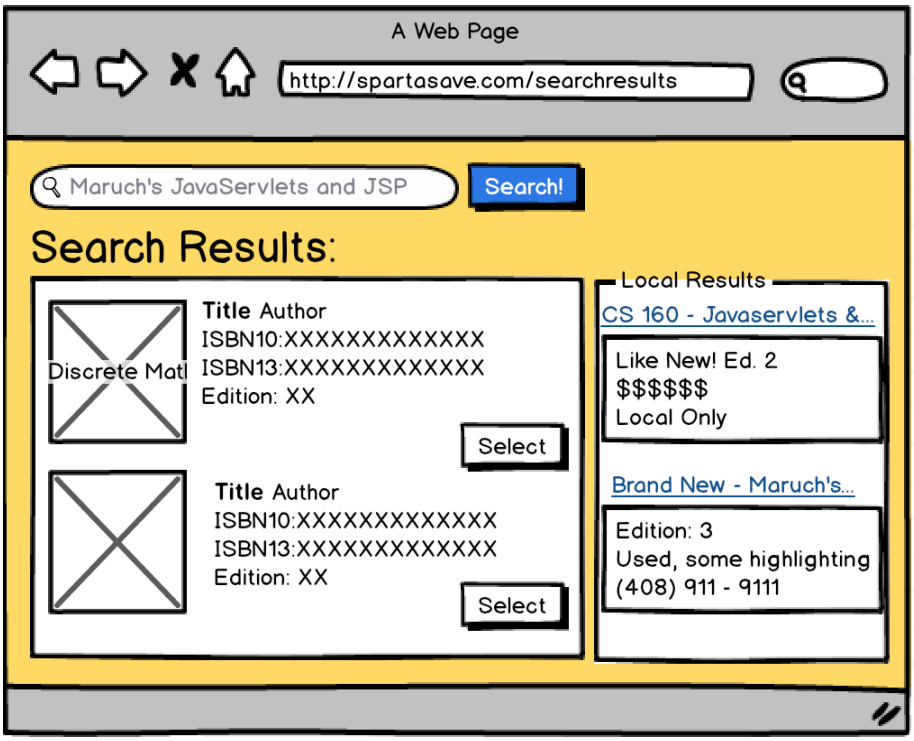


Figure 4. [SearchResults.jsp] User views search results from local postings & online retailers

As we moved forward with the design, we used Adobe Photoshop to create a modern layout for our responsive design using Bootstrap.js.

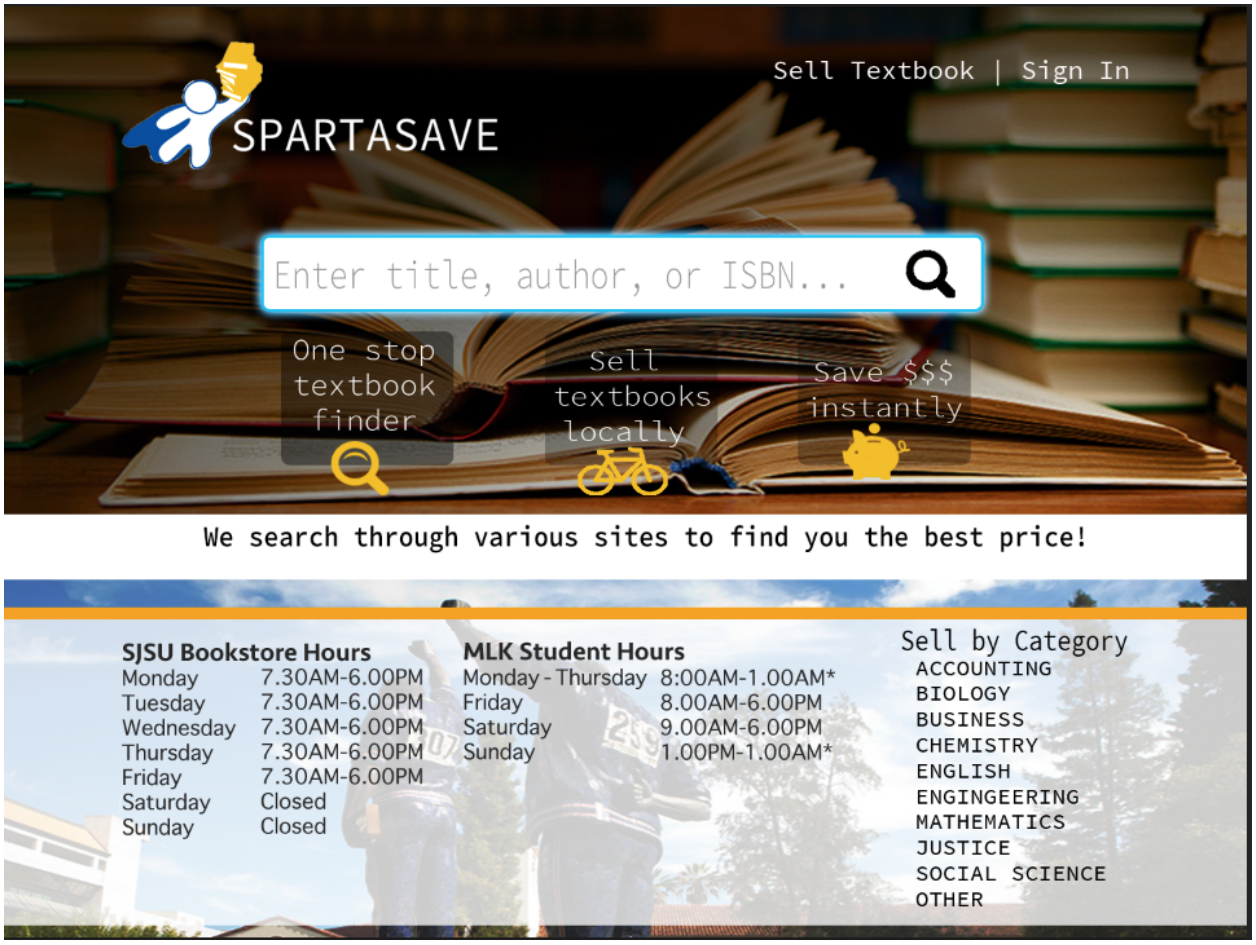


Figure 5. [index.psd] Mockup of view created on Photoshop before converting into jsp pages

Product Design

Software Architecture

UML package and class diagrams

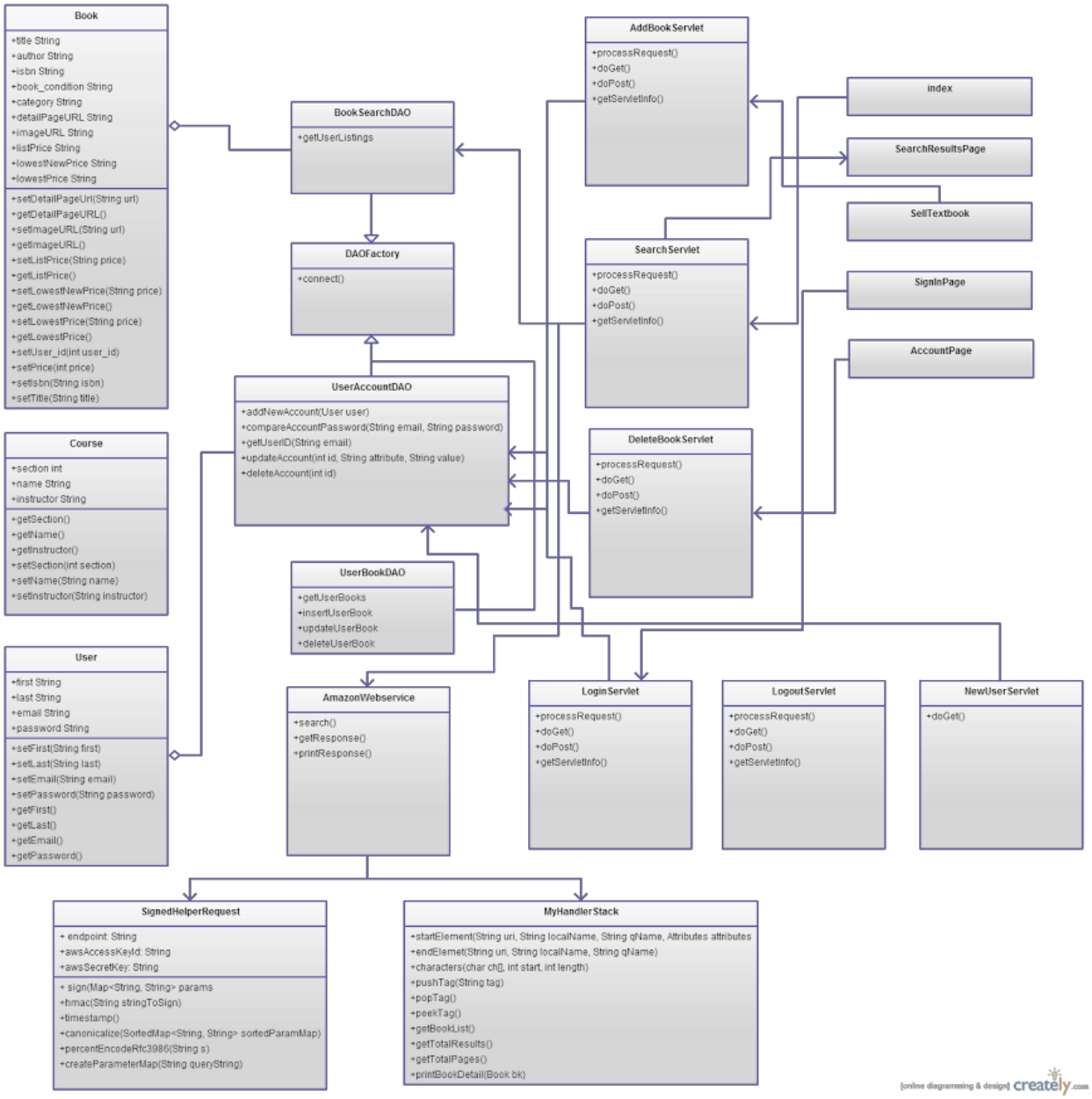


Figure 6. UML Diagram

Model-View-Controller class descriptions

As for the implementation of our product, we have implemented a Model-View-Controller (MVC) architecture. Our model consists of BookSearchDAO,

UserAccountDAO, UserAccountDAO, and AmazonWebservice. Our controllers are servlets and consists of AddBookServlet, SearchServlet, DeleteBookServlet, LoginServlet, LogoutServlet, and NewUserServlet. Our view is made of JSP pages, and include index, SearchResultsPage, SellTextBook, SignInPage, and AccountPage. The rest of our classes support the classes the make up the MVC.

The BookSearchDAO class is a java class that extends DAOFactory, which is a support class that connects our model to our database. BookSearchDAO can then query our database for specific book listings and return the results from the database in an ArrayList of Book objects, which is a format to organize information on books. Similarly, AmazonWebService also extends DAOFactory and queries the Amazon database using SignedHelperRequest and MyHandlerStack for specific book listings and returns the results as an ArrayList of Book objects. To handle changes to our database tables that hold information on user sold books, we have the UserBookDAO which extends DAOFactory and include methods such as insertUserBook(), updateUserBook(), and deleteUserBook(). Finally, our UserAccountDAO deals with our database tables that hold accounts. It verifies user login, adds new accounts, and delete user accounts.

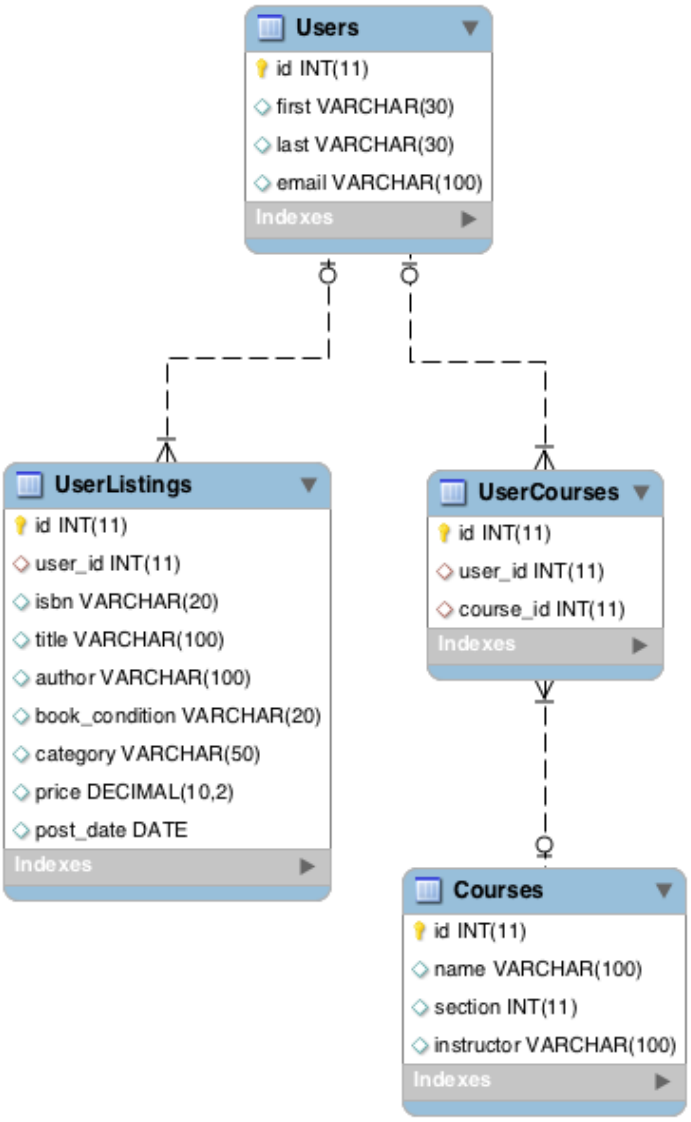
To connect our view with our model, we have our controllers, which are servlets. We have servlet classes for each major user request made from the view. SearchServlet takes inputted string from our view and processes HTTP requests. It calls methods from our model classes, such as BookSearchDAO and AmazonWebservice, to query given strings. AddBookServlet and DeleteBookServlet edit user listed books by connecting to the UserAccountDAO and UserBookDAO. LoginServlet and LogoutServlet handle user login and logout with Cookies and process login verification. NewUserServlet handles a request to create a new account by connecting to the UserAccountDAO.

To interact with our users, we have our view classes, which are JSP pages. Index is our main page which users are initially directed to when entering our application. It has a search bar when users can search for specific books. It connects to our SearchServlet to process queries for whatever is entered into the search bar. SearchResultsPage is where SearchServlet displays results given by the model classes. SellTextbook is our page that allows users to enter information about a textbook they want to sell. It connects to AddBookSevlet to enter the users new book into our database. SignIn page is where are users sign in to their accounts using their username and password. The username and password are then sent to be verified in UserAccountDAO through LoginServlet. Finally, our AccountPage is where our users can view and edit their account information and view and update books that they have posted to be sold. It connects to DeleteBookServlet to make changes to any of the books they have listed.

Data models

Logical: Entity-Relationship Diagrams

Our local database consists of four tables: Users, UserListings, UserCourses and Courses.
Figure 7. Entity - Relationship Diagram



Users

- Stores information about a user.
- Fields: id (primary key), first, last, and email
- Data will come from when the user is on SignInPage.jsp and creates a new account.

UserListings

- Books that users can post onto the website to sell.
- Fields: id (primary key), user_id (foreign key referencing Users table "id" field), isbn, title, author, book_condition, category, price, and post_date
- Data will come from when the user is on AccountPage.jsp and adds a new book to sell.

UserCourses

- Courses a user is taking.
- Fields: id (primary key), user_id (foreign key referencing Users table "id" field), course_id (foreign key referencing Courses table "id" field)
- Data will come from when the user is on AccountPage.jsp and fills out the form to add a new course they are currently enrolled in.

Courses

- Information about each course.
- Fields: id (primary key), name, section, instructor
- Data will be imported from SJSU course catalog

Physical: Database Table Diagrams

SQL Files

Please refer to

- create_spartasavedb.sql for SQL dump output file
- query-findseller.sql and query-categorysearch.sql for SQL queries

Sample Queries

1. Find users selling the book "Java Servlets"

```
SELECT first, last, book_condition, price
FROM Users, UserListings
WHERE user_id = Users.id
      AND title = 'Java Servlets';
```

Output:

```
mysql> SELECT first, last, book_condition, price
    -> FROM Users, UserListings
    -> WHERE user_id = Users.id
    -> AND title = 'Java Servlets';
+-------+-------+----------------+-------+
| first | last  | book_condition | price |
+-------+-------+----------------+-------+
| Kevin | Tan   | Like New       | 30.00 |
| Frida | Kahlo | Used           | 20.00 |
+-------+-------+----------------+-------+
2 rows in set (0.00 sec)
```

2. Find books in the "Computer Science" category

```
SELECT isbn, title, author, book_condition, price
FROM UserListings
WHERE category = 'Computer Science'
ORDER BY price;
```

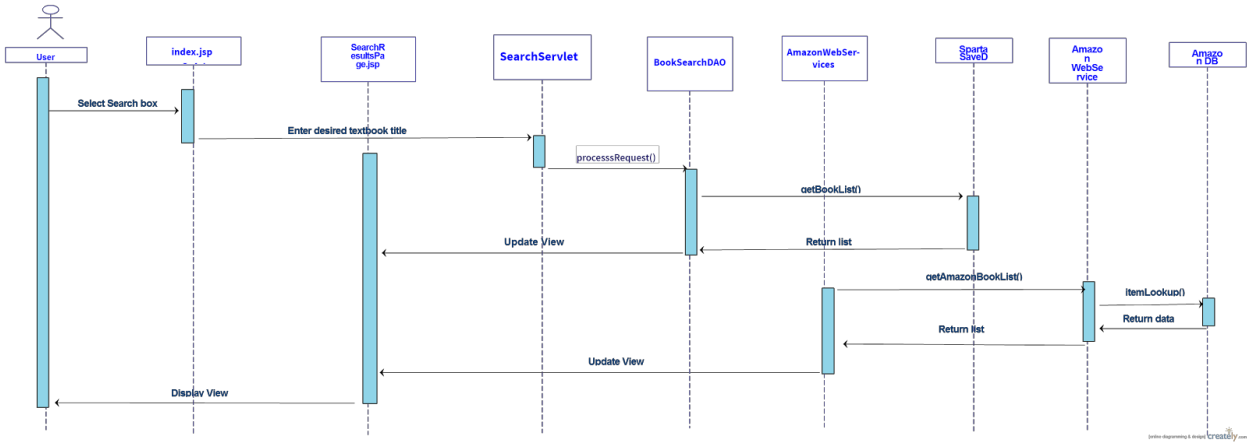
Output:

```
mysql> SELECT isbn, title, author, book_condition, price
    -> FROM UserListings
    -> WHERE category = 'Computer Science'
    -> ORDER BY price;
+---------------+-----------------------+----------------+----------------+-------+
| isbn          | title                 | author         | book_condition | price |
+---------------+-----------------------+----------------+----------------+-------+
| 9783147484236 | Object-Oriented Design | Greg Lock      | Good           | 10.00 |
| 9783161484100 | Java Servlets         | Michael Pearson | Used           | 20.00 |
| 9783161484100 | Java Servlets         | Michael Pearson | Like New       | 30.00 |
+---------------+-----------------------+----------------+----------------+-------+
3 rows in set (0.00 sec)
```

Dynamic behavior

UML Sequence and Statechart Diagrams

The following sequence diagram corresponds to our main feature - to search for a textbook. It is also attached for your reference [refer to file: SpartaSave_Sequence_Diagram.png].



Project Plan

Hardware and Software Requirements

In order to run SpartaSave, you will only need a charged laptop for hardware requirements. For software, you will need the following free software downloaded on your laptop:

- Netbeans IDE (https://netbeans.org/downloads/)
- Apache Tomcat Server - if not already included with Netbeans (http://tomcat.apache.org/download-70.cgi)
- MySQLWorkbench (http://dev.mysql.com/downloads/workbench/)
- A web browser (Firefox, Safari, Google Chrome, Internet Explorer)
- Access to our GitHub repository (https://github.com/dorathiendo/SJSU-CS160-QuietCoders/tree/master/SpartaSaveV2)

Project Schedule

We organized our tasks at hand by creating a Gantt Chart.

Gantt Chart

(refer to file: QuietCoders-GanttChart.gan)

Resource Usage

(refer to file: QuietCoders-GanttChart.gan)

Deployment

Production build and deployment scripts

(refer to files: build.xml and build.properties)

User Instructions

Figure 6. index.jsp

Creating a new user:

1. On index.jsp (Fig. 6), click on "Sign In" in the top-right.

2. On the bottom of the pop-up window, click "Sign Up!"
3. Enter in your first name, last name, email and password and click "Join."
4. You will be redirected to AccountPage.jsp and automatically signed in.

Signing In:

1. From the index page, click on "Sign In" on the top-right.
2. Enter in your email and password you previously created.
3. You will be redirected to AccountPage.jsp and be able to view textbook you have currently listed.

Adding a new textbook:

1. After signing in, click on "Sell Textbook" on the top-right.
2. Fill in all of the fields and click "Submit."
3. You will be redirected to AccountPage.jsp and the new textbook submitted should be added to the list of textbooks you are currently selling.

Searching a textbook:

1. In the search bar found on the middle top, enter in your search term, for example "Biology."
2. The search results will be returned, searching through user listings and amazon listings. (Fig. 7. below shows an example of a partial search results page returned after searching "Biology.")

Figure 7. Searching "Biology" Textbooks

Viewing Amazon page of a specific textbook:

1. After searching for a textbook and getting the search results page, click on the title of a specific textbook.
2. For example using (Fig. 7), clicking on "Campbell Biology (9th Edition)" will bring you to the amazon product page for that specific textbook.

Deleting a textbook:

1. On AccountPage.jsp, click on the "Delete" button under the "Manage" section next to the textbook you wish to delete.
2. The page will be refreshed and the textbook will be removed.

Signing Out:

1. On AccountPage.jsp, click on "Sign Out" on the top-right.
2. You will be redirected to the index.jsp and be logged out.

Release Notes

Initially, our feature list was quite ambitious for a team building a web application in a time span of two months and for the very first time. Originally, our feature listed included:

1. A search engine that allows students to find textbooks from various sites (i.e. Amazon, Chegg, etc.)
    a. Sorts by price, new or used
    b. Searches user postings as well
    c. List other editions of the same textbook and their prices
2. Users are able to sell their own textbooks and/or buy them from other users
    a. Create posting by category to sell books and remove the listing once it is sold
    b. Browse posted books by category
3. Registered users shall be able to create a running list of courses.
4. Internal messaging system
    c. Buyers and sellers can arrange details of the transaction

We approached each numbered item as a release candidate (RC). Therefore, our first priority was the search engine. As a result, we implemented a search engine that displays textbooks for sale from Amazon and from our local database. We were not able to finish the sorting by price and condition options, but it was a great accomplishment to get the Amazon integration working. Our second priority to allow students to sell and buy textbooks was also partially met. The user is able to sell their textbook based on category and remove the posting at anytime. However, a buy is not, at the moment, able to browse for textbooks based on category. The RC's we were not able to look into was the run on list of course list for users and internal messaging system for online users. However, in the time allotted, we believe this release is a personal achievement for us.