SJSU STUDENT BOOKSTORE

TEAM 6

Adrian Chan

Bikalp Timalsina

Jennifer Yang

PREPARED FOR

SUMMER 2020

CS157A Section 80

San Jose State University

Project Overview

The main purpose of this project is to develop a web application platform that can be used by San Jose State University to sell textbooks that are relevant to the students' courses. The web application platform called “SJSU Student Bookstore” will provide SJSU students a convenient, minimalist way to shop for college textbooks within the SJSU community.

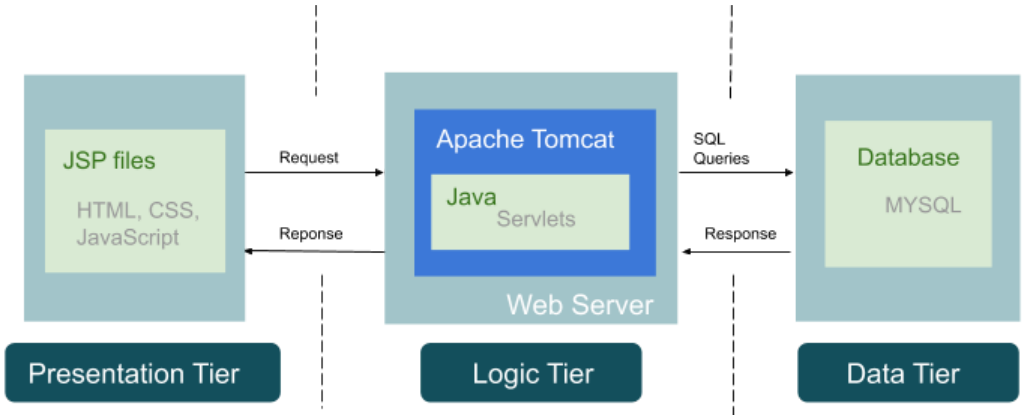
The Problem

San Jose State University has a place where they sell textbooks, called SJSU Spartan Bookstore. However, SJSU Spartan Bookstore sells more than just textbooks. SJSU Spartan Bookstore sells many different items, ranging from makeup to laptops. Instead of a bookstore , SJSU Spartan Bookstore is more of a marketplace. People can buy miscellaneous items like umbrellas, instant ramens, SJSU clothing merchandise, etc. Aside from textbooks, SJSU Spartan Bookstore has reading for fun novels and magazines. Anyone can buy items from SJSU Spartan Bookstore and SJSU Spartan Bookstore offers some great discounts. However, most discounts are for SJSU clothing merchandise.

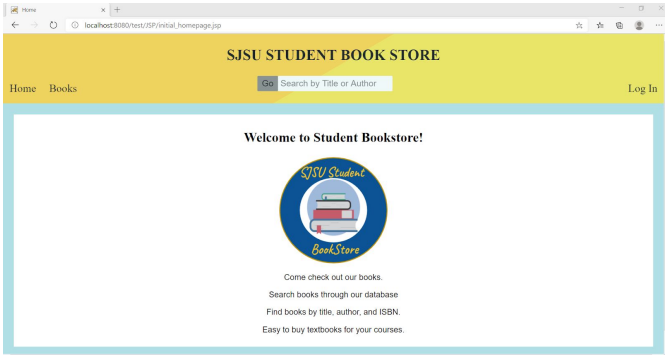
Why ‘SJSU Student Bookstore’

SJSU Student Bookstore solely focuses on selling textbooks used in San Jose State University courses and is dedicated for SJSU students. SJSU Student Bookstore will offer textbooks at discounted price as a special treat for the students . Students will have a convenient, minimalist way to shop for relevant textbooks for their classes.

System Requirements



Presentation Tier



In SJSU Student Bookstore, the user interface will be a web application that handles the interaction with users and displays information. The front-end of the web application will use JSP pages with HTML, CSS, and JavaScript.

Logic Tier

SJSU Student Bookstore will use Java code as a backend server for handling user requests and communicating with the database.

Data Tier

MySQL will be used as the Relational Database Management System to manage and access data.

Revised Functional Requirements

Users

The users of SJSU Student Bookstore will be SJSU students and SJSU admins. Only SJSU students will be able to purchase textbooks and share textbooks with other SJSU students. There will be a book display for non-SJSU students. However, non-SJSU students will not be able to purchase or share textbooks. SJSU admins manage the students and textbooks.

Functionality and Features

SJSU Student Bookstore will have a login page where students can sign in (username, password). With valid university credentials (SJSU id), students can sign up for an account. Students can browse, purchase and share textbooks among other students. Admin can add and delete books/members from the SJSU Student Bookstore database and see data of books and members.

Input/Outputs

The inputs for SJSU Student Bookstore would include the entered student data, admin data and book data. The projected outputs would be the ability to view books and manage books/members from the database and display on the user interface.

Functional Requirements

Search

- The user can query the database system using a search bar.
- The user can search by title or author
- The database would return a list of results and the list would be displayed on the webpage for the user.
- If no results are available, the webpage would request the user to enter in another search.
- Any user/guest has access to this function.

Share

- The users can share books to one another, users have to login to their account in order to use this functionality.
- Shared items can only be viewable between the sender and receive..
- There will be a textfield which will allow the users to enter what book and who the users they would like to share it with.

Display

- The display function provides an easy way for users to browse in the database. There will be a page where it will provide users with every book in the database and they can browse, read the book summary. This function helps users to choose books by browsing.
- All books will be stored in one database so it will be easy for the users to browse using the display function.

Most Recently Searched

- This allows the user to see their top 5 most recent book searches.
- This will be done by storing the name of the search into the database each time the user searches a book.

Account-Create User

- The user will be given the option to login or sign up.

- Once the account is created, each user will have their own database, which will have their own personal information. These would be like the books they rented, bought, etc.

Purchase History

- When the user checks out thier cart they are able to view the books on the account page.
- From the checkout cart, the book data will be stored in the purchase datatable.

Add/Remove Book in Checkout

- When in the checkout cart users can have the option to delete books which they don't want anymore.
- When the user likes a book they want to buy, they can click the add to cart function which would be stored in the checkout cart. They can choose multiple books.

Add/Remove Book in Database

- The admin user has control of what books can be added to/deleted from the SJSU Student Bookstore database.

Remove Member

- The admin user has control of removing members from the SJSU Student Bookstore database.

Request Books

- Users can request books to be added to the SJSU Student Bookstore. Admin will be able to see who and what books are requested.

Signout

- At the end the users can sign out from their account once they are done.
- After signing out, the user will be redirected to the guest webpage.

Non Functional Requirements

Non-functional requirements contain many aspects like accessibility, security, etc. With the time of CS157A course, Team 6's goal is to build a three-tier architecture application that provides students a minimalist, convenient way to browse and purchase books in the SJSU community.

GUI

- Our Graphical User Interface(GUI) will be designed following certain principles of Human-Computer Interactions(HCI) to achieve the convenient, minimalist display. Our application will strive for consistency, in terms of usability, colors, etc. When designing the application, we will follow Norman's 7 principles.
- For example, the search bar will be at the top of the application. The search bar is a main component of our application.
- In accordance with Norman's principles, core user functions are clearly visible. The members account and checkout button will be on the left side of the application to maintain good mapping.

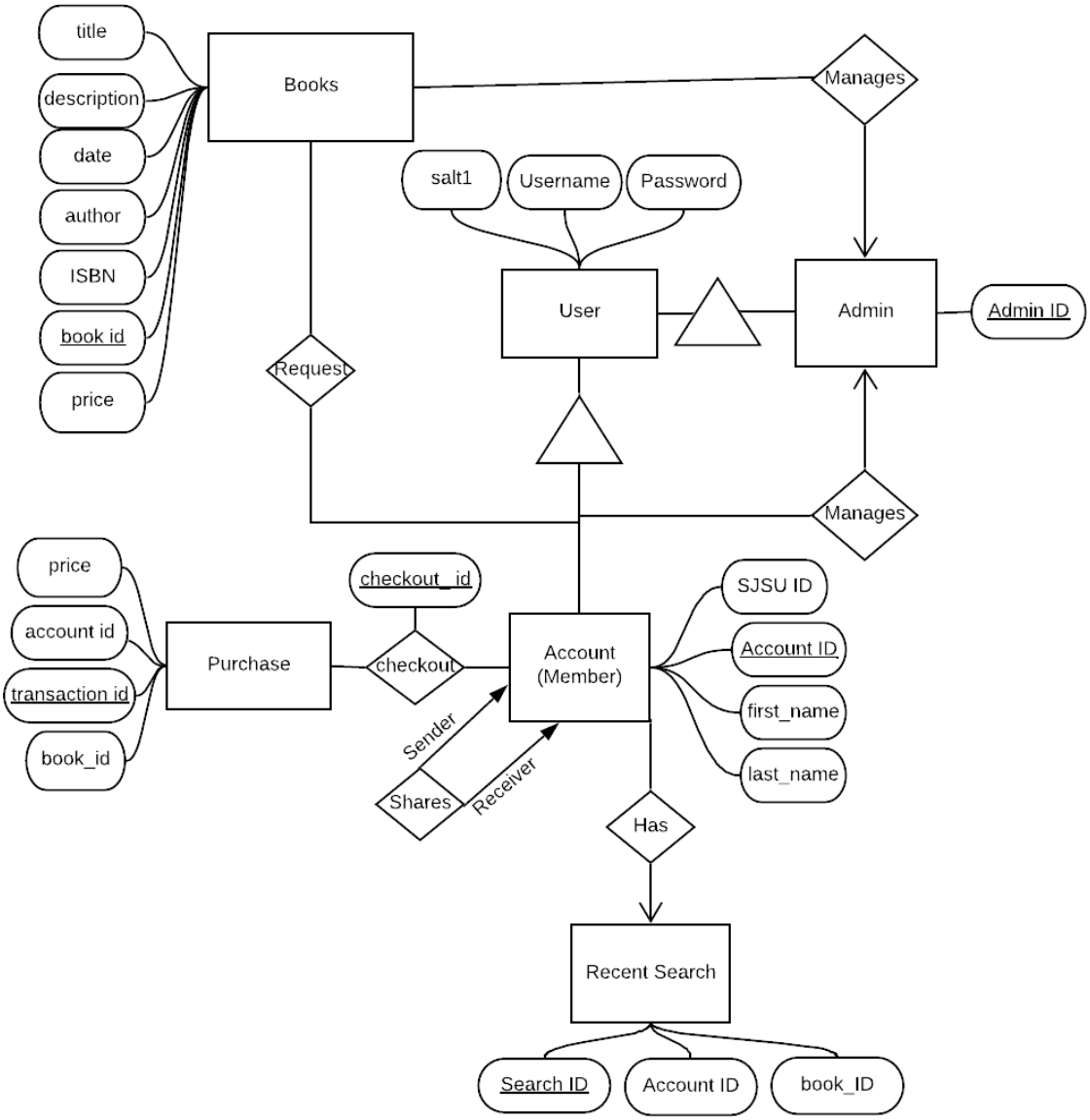
Security

- Users are asked to choose their username and password. Their information will be saved to the database.
- When the users login they have to enter the username and password which they first created.
- For any text related fields in the website there will be security to prevent SQL injection.
- The users password will be hashed and salted, for security and stored into the database.

Access control

- Each user will have their own accounts giving them access to share, checkout, sell, and discover books that may interest them.
- When logged in, users can recommend a book to another valid user and also upload a book that they want to sell to the database.
- Users with no account do not have access to recommending a book or uploading a book they want to sell in the database system.
- The highest level of permission is granted to the admins or one.sjsus security team.
- The admin team would have control of the entire database system while only the users can insert books they want to sell. Anyone who visits the web page can have access to the search function.

E/R Diagram


title
description
date
author
ISBN
book id
price
Books
Manages
salt1
Username
Password
User
Admin
Admin ID
Request
Manages
price
account id
transaction id
book_id
Purchase
checkout_ id
checkout
Account
(Member)
SJSU ID
Account ID
first_name
last_name
Sender
Receiver
Shares
Has
Recent Search
Search ID
Account ID
book_ID

Description

Account

Account user is an entity that represents the SJSU student member. A member must have a sjsu_id, account_id, username, password. The account_id is the primary key for the User table.

Books

Books is an entity that represents the books in SJSU TextBookStore.A book has title, description, date, author, book_id, isn, and price. The book_id is the primary key for the Books table.

Admin

Admin is an entity that manages members and books in the system. Admin has a username, password, and a primary key admin_id.

Purchase

Purchase is an entity that represents the purchases made by a member. Purchase has books_id, account_id, total_amount, and transaction_id. The primary key is transaction_id.

Recent_search

Recent_search is an entity that displays the five most recently searched books in the system for a member. Recent_search has Time_stamp, search_id, and account_id. The Time_stamp is the primary key for the table Recent_search.

Members_make_purchase

Members_make_purchase is a relationship that indicates the member made how many purchases. Each member can make one or more purchases.

Members_request_books

Members_request_books is a relationship that connects the Members and Book entity. Each member can request one or more books.

Members_shares_member

Members_shares_member is a relationship within the Members. The sender member can share a book to the receiver member. Members can share one or more books to other members.

Members_has_recent_search

Members_has_recent_search is a relationship that connects Members and recent_search entity sets. Each member has one recent search. Each Recent Search can have zero to many members.

Admin_manages_books

Admin_manages_books is a relationship that connects Admin and Books entity sets. Admin can manage zero to many books. Each book gets managed by one Admin.

Admin_manages_members

Admin_manages_members is a relationship that connects Admin and Members entity sets. Admin can manage zero to many members. Each Member gets managed by one Admin.

SJSU Student Bookstore Schema

Database Name: SJSU_textbookstore

Database: Relationship

1. Member shares Members
    a. Member(Account_id, Account_id)
2. Member request Books
    a. Member(Account_id, Book_id)
3. Member checkout Books
    a. Member(Checkout_id, Account_id, Book_id)
4. Admin manages Members
    a. Admin(Admin_ID, Account_id)
5. Admin manages Book
    a. Admin(Admin_ID, book_id)

Database: Entity Set

1. Books(title, description, date, author, book_id, isn, price)
2. Members(password, username,sjsu_id, account id)
3. Recent Search(account_id, search_id, timestamp)
4. Purchase(transaction_id, account_id, price, books_id)
5. Admin(Username, Password, Admin ID)

Tables

ACCOUNT

```
mysql> describe account;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| account_id | int          | NO   | PRI | NULL    | auto_increment |
| sjsu_id    | varchar(250) | NO   | PRI | NULL    |                |
| user_name  | varchar(250) | NO   | UNI | NULL    |                |
| first_name | varchar(250) | NO   |     | NULL    |                |
| last_name  | varchar(250) | NO   |     | NULL    |                |
| salt1      | varchar(250) | NO   |     | NULL    |                |
| password   | varchar(250) | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.account;
+------------+-----------+-----------+------------+-----------+----------+------------------------------------------------------------------+
| account_id | sjsu_id   | user_name | first_name | last_name | salt1    | password                                                         |
+------------+-----------+-----------+------------+-----------+----------+------------------------------------------------------------------+
|          1 | 1003265746 | King      | King       | King      | 552973589 | 1ad9ad935a98d1ccc783a4c59a46aa1bd722836e0bfa3403bc7fa18d8b308f2c |
|          2 | 1003256454 | Natsu     | Natsu      | Dragneel  | 618531137 | cae9f4c8a371f16ca105f8ca1d0430f265bd3483c112a457d578a8a6d6fa033a |
|          3 | 1005623232 | Lucy      | Lucy       | Heartgilia | 41019831 | d573e8b8f6d24a1a15aee4c741c9458a8cbbbd1e9efd891145e8da1fbbdb2eed |
|          4 | 1009876545 | John      | John       | Cena      | 763061970 | e2c567243a1d77f6472fc8055ab75b9afbfb787bf241be4477d3e15ae9e5fa27 |
|          5 | 1005646854 | Jo        | Jo         | Stewart   | 393108051 | 4ac8b4b603208f3d716b3b687025de6802ebee97b7806a9532c1404eec7afbfc |
|          6 | 1006546546 | Humphrey  | Humphrey   | Manning   | 990296907 | af6ec5ef05ce477f3a8a3400befe1a5b70c7d65c0abf08725e23066183ade536 |
|          7 | 1006573545 | Reef      | Reef       | Corted    | 645112129 | 26dad7f8b5cefb06bc777a6d906b2da29aa2d53fc266b35d0f186505f170f5ef |
|          8 | 1006576854 | Erin      | Erin       | Ho        | 435689532 | a2ef7c70b1cfb9a9821f372d010991d9c323f3542942d05f50a739590643afe7 |
|          9 | 1006576584 | Ella      | Ella       | Mcdermott | 977259056 | ae5b91f14e49ed46b6f6ee57195cc9bd598b038d227c53a46ae8a40008f96589 |
|         10 | 1004345615 | Davina    | Davina     | Atkinson  | 146955068 | 27dfa310374faf05582cfbc638fd95a7f8a69fd4f70c8d279d7752aae6df8ff2 |
|         11 | 1006547654 | Yasser    | Yasser     | Guthrie   | 30303045  | c5865457c5e8796dcc7dd0f8552f4c3d49d3d166ac065f0c6fc8f6485c845684 |
|         12 | 1006546875 | Valentina | Valentina  | Whittaker | 619375957 | c9c7cca2b7462d02f38a5ee6806a6e2e42231648bac24867cbb6839a6c43a62a |
|         13 | 1002343545 | Robin     | Robin      | Nico      | 243634916 | 0720269605853 96afe5c122c93517b8a6c17c262818d211088ea10d0db981f7b |
|         14 | 1005673326 | Sanji     | Sanji      | Vinsmoke  | 66317255  | 520a9296aaa1d519c5dfe2a734d9da8b91c465569f4f101688c616c31904acac |
|         15 | 1003554654 | Doflamingo | Doflamingo | Donquixote | 579845303 | f96731430314db1dbbf5581c4f1ba1030055c589067b02ce5be6ad72198c3295 |
|         16 | 1005646512 | Sky       | Sky        | Woodson   | 740290531 | 523e35cb0ba6bf705be28d16ae342486d4f5690d89faa73826deb4b4b9b620a4 |
+------------+-----------+-----------+------------+-----------+----------+------------------------------------------------------------------+
16 rows in set (0.00 sec)
```

ADMIN

```
mysql> describe admin;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| admin_id  | int          | NO   | PRI | NULL    | auto_increment |
| user_name | varchar(250) | NO   | UNI | NULL    |                |
| salt1     | varchar(250) | NO   |     | NULL    |                |
| password  | varchar(250) | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.admin;
+----------+-----------+-----------------+------------------------------------------------------------------+
| admin_id | user_name | salt1           | password                                                         |
+----------+-----------+-----------------+------------------------------------------------------------------+
|        1 | John      | aadsf43523      | e810f5170bd2fa70cf5ce5f609448ee9ea514270f6150005edef71622a0289ce |
|        2 | Kite      | 313a1dsfa       | e242f043caac8954a3eb6899cf4c145662960198f3d9b8468e6093cf356689eb |
|        3 | Silvya    | 4645afdsf       | c39ebdef4c7eda44e36f8dddab329ae0b757cfcea02c300cfdb0f5f85662ee45 |
|        4 | Blue      | 1324qererasdf   | b2d5429cd2f1686b1f95390c772f1fb009137216350a630dd3106e847c1f5e3c |
|        5 | Green     | asdfasdfa2324   | 0fa21c9f297dc4c4827b06d0df6e0b604edb6e3a384b2138583438790a81b8b1 |
|        6 | Apple     | asdf234         | 4ed36068fc960acfdde41141928c4400633234428e0fb4806f6ca850efa77f5b |
|        7 | August    | 253132a1sdf     | 9efe546b85e349ab809dfa11937d31fd53d576581e8daacac4d03dc5d8649f44 |
|        8 | September | asdf234243      | 2ec6d3a9251059839883a671238f7a19eec403700db8f50427e1ff52bd16228e |
|        9 | Tommy     | asdfafd345345345 | a2c208e79c1d503130a4e608a815553ea0c14f4abda2c61cb2e6a0d4881ba6dc |
|       10 | Bob       | 3123213132qer   | 4dd1d6a14582d49a3a668c204a1fe1eceef7eae731357dd5aacaae70104189db |
|       11 | Syam      | asdfa453545435  | fb68222ccaf1f49f03b5faecbea9da3d52e77442eae012b23a05409bfb6d1b91 |
|       12 | Krish     | 654a65sdf16afd  | 56c35e33e1ef308be824974242df3c7946b0ec732b63d615218854d454658028 |
|       13 | Jack      | 54654bjkhabdf   | c37770502587af3aa0a9b2d916f81ed08b47276a98edded7455d8b0a7e72e823 |
|       14 | Daniel    | 45465asdfasdf   | 827383c7d4b4522cadd4c91c1a63082c3207261f8ec24e51402cf0df30aa316f |
|       15 | Regular   | asdrtyryt       | ef236b5d08c735c331e707723b37a10c3c532ba869123f78af0603335e9a7054 |
+----------+-----------+-----------------+------------------------------------------------------------------+
15 rows in set (0.00 sec)
```

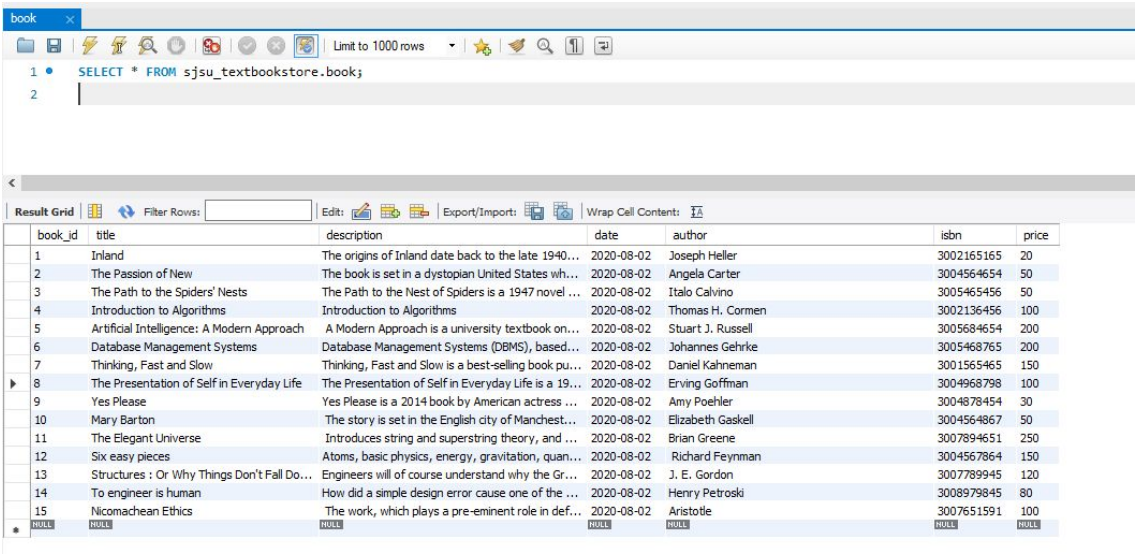
ADMIN_BOOKS

```
mysql> describe admin_books;
+----------+------+------+-----+---------+-------+
| Field    | Type | Null | Key | Default | Extra |
+----------+------+------+-----+---------+-------+
| admin_id | int  | NO   | PRI | NULL    |       |
| book_id  | int  | NO   | PRI | NULL    |       |
+----------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.admin_books;
+----------+---------+
| admin_id | book_id |
+----------+---------+
|        1 |       1 |
|        1 |       2 |
|        1 |       3 |
|        1 |       4 |
|        1 |       5 |
|        2 |       6 |
|        2 |       7 |
|        2 |       8 |
|        2 |       9 |
|        2 |      10 |
|        3 |      11 |
|        3 |      12 |
|        3 |      13 |
|        3 |      14 |
|        3 |      15 |
+----------+---------+
15 rows in set (0.00 sec)
```

ADMIN_Members

```
mysql> describe admin_members;
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| admin_id   | int  | NO   | PRI | NULL    |                |
| account_id | int  | NO   | PRI | NULL    | auto_increment |
+------------+------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.admin_members;
+----------+------------+
| admin_id | account_id |
+----------+------------+
|        3 |          1 |
|        2 |          2 |
|        3 |          3 |
|        1 |          4 |
|        1 |          5 |
|        2 |          6 |
|        1 |          7 |
|        2 |          8 |
|        3 |          9 |
|        3 |         10 |
|        3 |         11 |
|        3 |         12 |
|        2 |         13 |
|        1 |         14 |
|        3 |         15 |
|        1 |         16 |
+----------+------------+
16 rows in set (0.00 sec)
```

Book



```
SELECT * FROM sjsu_textbookstore.book;
```

book_id	title	description	date	author	isbn	price
1	Inland	The origins of Inland date back to the late 1940...	2020-08-02	Joseph Heller	3002165165	20
2	The Passion of New	The book is set in a dystopian United States wh...	2020-08-02	Angela Carter	3004564654	50
3	The Path to the Spiders' Nests	The Path to the Nest of Spiders is a 1947 novel ...	2020-08-02	Italo Calvino	3005465456	50
4	Introduction to Algorithms	Introduction to Algorithms	2020-08-02	Thomas H. Cormen	3002136456	100
5	Artificial Intelligence: A Modern Approach	A Modern Approach is a university textbook on...	2020-08-02	Stuart J. Russell	3005684654	200
6	Database Management Systems	Database Management Systems (DBMS), based...	2020-08-02	Johannes Gehrke	3005468765	200
7	Thinking, Fast and Slow	Thinking, Fast and Slow is a best-selling book pu...	2020-08-02	Daniel Kahneman	3001565465	150
8	The Presentation of Self in Everyday Life	The Presentation of Self in Everyday Life is a 19...	2020-08-02	Erving Goffman	3004968798	100
9	Yes Please	Yes Please is a 2014 book by American actress ...	2020-08-02	Amy Poehler	3004878454	30
10	Mary Barton	The story is set in the English city of Manchest...	2020-08-02	Elizabeth Gaskell	3004564867	50
11	The Elegant Universe	Introduces string and superstring theory, and ...	2020-08-02	Brian Greene	3007894651	250
12	Six easy pieces	Atoms, basic physics, energy, gravitation, quan...	2020-08-02	Richard Feynman	3004567864	150
13	Structures : Or Why Things Don't Fall Do...	Engineers will of course understand why the Gr...	2020-08-02	J. E. Gordon	3007789945	120
14	To engineer is human	How did a simple design error cause one of the ...	2020-08-02	Henry Petroski	3008979845	80
15	Nicomachean Ethics	The work, which plays a pre-eminent role in def...	2020-08-02	Aristotle	3007651591	100

Checkout

```
mysql> describe checkout;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| checkout_id | int          | NO   | PRI | NULL    | auto_increment |
| book_id     | int          | NO   | PRI | NULL    |                |
| account_id  | int          | NO   | PRI | NULL    |                |
| title       | varchar(250) | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.checkout;
+-------------+---------+------------+-----------------------------------------+
| checkout_id | book_id | account_id | title                                   |
+-------------+---------+------------+-----------------------------------------+
|          20 |       5 |         14 | Artificial Intelligence: A Modern Approach |
|          21 |       2 |         14 | The Passion of New                      |
|          22 |      10 |         14 | Mary Barton                             |
|          26 |      11 |          9 | The Elegant Universe                    |
|          27 |      15 |          9 | Nicomachean Ethics                      |
|          30 |      12 |         11 | Six easy pieces                         |
|          31 |       8 |         11 | The Presentation of Self in Everyday Life |
|          32 |       9 |         11 | Yes Please                              |
|          36 |       8 |          5 | The Presentation of Self in Everyday Life |
|          37 |       1 |          5 | Inland                                  |
|          38 |       9 |          5 | Yes Please                              |
|          39 |       7 |          5 | Thinking, Fast and Slow                 |
|          40 |       2 |          5 | The Passion of New                      |
|          48 |       6 |          7 | Database Management Systems             |
|          49 |       8 |          7 | The Presentation of Self in Everyday Life |
+-------------+---------+------------+-----------------------------------------+
15 rows in set (0.00 sec)
```

Member Share

```
mysql> describe member_share;
+-------------+------+------+-----+---------+----------------+
| Field       | Type | Null | Key | Default | Extra          |
+-------------+------+------+-----+---------+----------------+
| share_id    | int  | NO   | PRI | NULL    | auto_increment |
| sender_id   | int  | NO   |     | NULL    |                |
| book_id     | int  | NO   | PRI | NULL    |                |
| receiver_id | int  | NO   |     | NULL    |                |
+-------------+------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.member_share;
+----------+-----------+---------+-------------+
| share_id | sender_id | book_id | receiver_id |
+----------+-----------+---------+-------------+
|        1 |         1 |       1 |           1 |
|        2 |         3 |      11 |          13 |
|        3 |         3 |      11 |          13 |
|        4 |        14 |       7 |           3 |
|        5 |        11 |       8 |           1 |
|        6 |        11 |       4 |           3 |
|        7 |        11 |      12 |           5 |
|        8 |         5 |       2 |          14 |
|        9 |         5 |      12 |           2 |
|       10 |         5 |      11 |          13 |
|       11 |         7 |       8 |           5 |
|       12 |         7 |       8 |           1 |
|       13 |         7 |       8 |           2 |
|       14 |         7 |       8 |          14 |
|       15 |         7 |       8 |           8 |
+----------+-----------+---------+-------------+
15 rows in set (0.00 sec)
```

Purchase

```
mysql> describe purchase;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| transaction_id | int          | NO   | PRI | NULL    | auto_increment |
| account_id     | int          | NO   | PRI | NULL    |                |
| book_id        | int          | NO   | PRI | NULL    |                |
| price          | varchar(250) | NO   |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.purchase;
+----------------+------------+---------+-------+
| transaction_id | account_id | book_id | price |
+----------------+------------+---------+-------+
|              1 |          1 |      10 | 50    |
|              2 |          1 |      12 | 150   |
|              3 |          2 |       2 | 50    |
|              4 |          3 |       2 | 50    |
|              5 |         14 |       6 | 200   |
|              6 |         14 |       9 | 30    |
|              7 |          9 |      11 | 250   |
|              8 |          9 |       9 | 30    |
|              9 |         11 |       8 | 100   |
|             10 |          5 |      10 | 50    |
|             11 |          5 |      15 | 100   |
|             12 |          7 |       4 | 100   |
|             13 |          7 |       8 | 100   |
|             14 |          7 |      15 | 100   |
|             15 |          7 |       2 | 50    |
+----------------+------------+---------+-------+
15 rows in set (0.00 sec)
```

Recent Search

```
mysql> describe recent_search;
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| search_id  | int  | NO   | PRI | NULL    | auto_increment |
| account_id | int  | NO   | PRI | NULL    |                |
| book_id    | int  | NO   | PRI | NULL    |                |
+------------+------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.recent_search;
+-----------+------------+---------+
| search_id | account_id | book_id |
+-----------+------------+---------+
|         1 |          1 |       1 |
|         2 |          1 |       9 |
|         3 |          1 |       4 |
|         4 |          1 |      11 |
|         5 |          1 |       5 |
|         6 |          1 |       7 |
|         7 |          2 |       8 |
|         8 |          2 |      14 |
|         9 |          3 |       9 |
|        10 |          3 |      10 |
|        11 |          3 |       2 |
|        12 |         14 |       2 |
|        13 |         14 |       7 |
|        14 |         11 |       8 |
|        15 |          5 |      14 |
+-----------+------------+---------+
15 rows in set (0.00 sec)
```

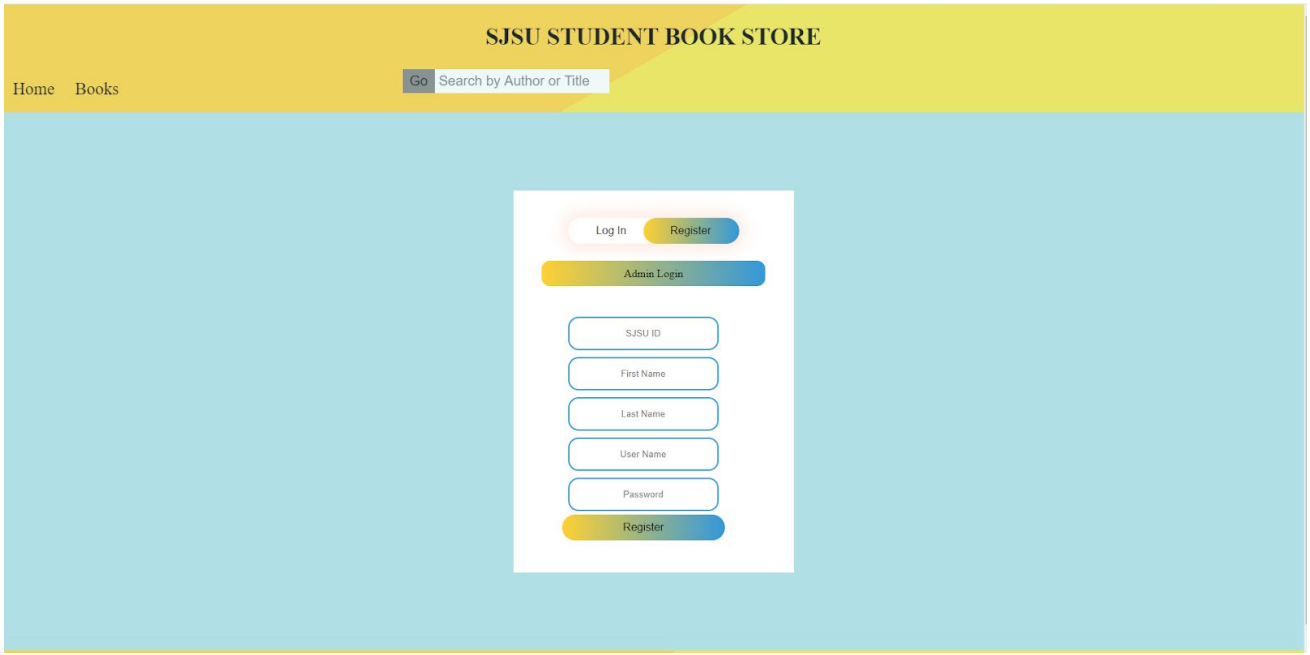
Request Books

```
mysql> describe request_books;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| request_id | int          | NO   | PRI | NULL    | auto_increment |
| account_id | int          | NO   | PRI | NULL    |                |
| title      | varchar(250) | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.request_books;
+------------+------------+---------------------------------+
| request_id | account_id | title                           |
+------------+------------+---------------------------------+
|          1 |          1 | Tao Te Ching                    |
|          2 |          5 | Meditations                     |
|          3 |          5 | Nicomachean Ethics              |
|          4 |          2 | The Art of War                  |
|          5 |          2 | Trainspotting                   |
|          6 |          2 | Gravity's Rainbow               |
|          7 |         14 | The Odyssey                     |
|          8 |         14 | The Lord of the Rings Trilogy   |
|          9 |         14 | Dubliners                       |
|         10 |          3 | Jurassic Park                   |
|         11 |          3 | One Flew Over the Cuckoo?s Nest |
|         12 |          3 | The Road                        |
|         13 |          7 | Fahrenheit 451                  |
|         14 |          7 | A Confederacy of Dunces         |
|         15 |          7 | Hard Times                      |
+------------+------------+---------------------------------+
15 rows in set (0.00 sec)
```

Implementation

ACCOUNT

Webpage



The above screenshot shows the main login/signup page where the user is first taken. If the user is not registered then they can signup or login if they are already registered.

Table

```
mysql> describe account;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| account_id | int          | NO   | PRI | NULL    | auto_increment |
| sjsu_id    | varchar(250) | NO   | PRI | NULL    |                |
| user_name  | varchar(250) | NO   | UNI | NULL    |                |
| first_name | varchar(250) | NO   |     | NULL    |                |
| last_name  | varchar(250) | NO   |     | NULL    |                |
| salt1      | varchar(250) | NO   |     | NULL    |                |
| password   | varchar(250) | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.account;
+------------+------------+------------+------------+------------+----------+------------------------------------------------------------------+
| account_id | sjsu_id    | user_name  | first_name | last_name  | salt1    | password                                                         |
+------------+------------+------------+------------+------------+----------+------------------------------------------------------------------+
|          1 | 1003265746 | King       | King       | King       | 552973589 | 1ad9ad935a98d1ccc783a4c59a46aa1bd722836e0bfa3403bc7fa18d8b308f2c |
|          2 | 1003256454 | Natsu      | Natsu      | Dragneel   | 618531137 | cae9f4c8a371f16ca105f8ca1d0430f265bd3483c112a457d578a8a6d6fa033a |
|          3 | 1005623232 | Lucy       | Lucy       | Heartgilia | 41019831  | d573e8b8f6d24a1a15aee4c741c9458a8cbbbd1e9efd891145e8da1fbbdb2eed |
|          4 | 1009876545 | John       | John       | Cena       | 763061970 | e2c567243a1d77f6472fc8055ab75b9afbfb787bf241be4477d3e15ae9e5fa27 |
|          5 | 1005646854 | Jo         | Jo         | Stewart    | 393108051 | 4ac8b4b603208f3d716b3b687025de6802ebee97b7806a9532c1404eec7afbfc |
|          6 | 1006546546 | Humphrey   | Humphrey   | Manning    | 990296907 | af6ec5ef05ce477f3a8a3400befe1a5b70c7d65c0abf08725e23066183ade536 |
|          7 | 1006573545 | Reef       | Reef       | Corted     | 645112129 | 26dad7f8b5cefb06bc777a6d906b2da29aa2d53fc266b35d0f186505f170f5ef |
|          8 | 1006576854 | Erin       | Erin       | Ho         | 435689532 | a2ef7c70b1cfb9a9821f372d010991d9c323f3542942d05f50a739590643afe7 |
|          9 | 1006576584 | Ella       | Ella       | Mcdermott  | 977259056 | ae5b91f14e49ed46b6f6ee57195cc9bd598b038d227c53a46ae8a40008f96589 |
|         10 | 1004345615 | Davina     | Davina     | Atkinson   | 146955068 | 27dfa310374faf05582cfbc638fd95a7f8a69fd4f70c8d279d7752aae6df8ff2 |
|         11 | 1006547654 | Yasser     | Yasser     | Guthrie    | 30303045  | c5865457c5e8796dcc7dd0f8552f4c3d49d3d166ac065f0c6fc8f6485c845684 |
|         12 | 1006546875 | Valentina  | Valentina  | Whittaker  | 619375957 | c9c7cca2b7462d02f38a5ee6806a6e2e42231648bac24867cbb6839a6c43a62a |
|         13 | 1002343545 | Robin      | Robin      | Nico       | 243634916 | 07202696058539 6afe5c122c93517b8a6c17c262818d211088ea10d0db981f7b |
|         14 | 1005673326 | Sanji      | Sanji      | Vinsmoke   | 66317255  | 520a9296aaa1d519c5dfe2a734d9da8b91c465569f4f101688c616c31904acac |
|         15 | 1003554654 | Doflamingo | Doflamingo | Donquixote | 579845303 | f96731430314db1dbbf5581c4f1ba1030055c589067b02ce5be6ad72198c3295 |
|         16 | 1005646512 | Sky        | Sky        | Woodson    | 740290531 | 523e35cb0ba6bf705be28d16ae342486d4f5690d89faa73826deb4b4b9b620a4 |
+------------+------------+------------+------------+------------+----------+------------------------------------------------------------------+
16 rows in set (0.00 sec)
```

Frontend Code

```
56
57      <!-- Display information retrieved from database into a table-->
58      <table class="table-desgin-two">
59          <tr>
60              <th>SJSU ID</th>
61              <th>Username</th>
62              <th>First Name</th>
63              <th>Last Name</th>
64          </tr>
```

This screenshot creates a table with four columns. The html for the data is embedded inside the backend code while retrieving from the account table.

Backend Code

```
<%
//Database retrieves the username
try{
    Connection connection =
    DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false","root", "N00bcakes");
    String account_id=(String)request.getSession().getAttribute("userid");
    String query = "SELECT first_name, last_name from account where account_id = ?";

    PreparedStatement prepared = connection.prepareStatement(query);
    prepared.setString(1, account_id);
    ResultSet rs = prepared.executeQuery();
    rs = prepared.executeQuery();
    while(rs.next()){
    out.println("<h1>Welcome " + rs.getString("first_name") + " " + rs.getString("last_name") + "</h1>");
    }
} catch(SQLException e) {
    out.println("SQLException caught: " + e.getMessage());
}
```

This above screenshot fetches a welcome message with the first and last name retrieved from the account table.

```
public class AccountDAO {
//  static PreparedStatement preparedStatement;
    public int register(Account acc) throws Exception {
        Random random = new Random();
        String salt1 = Integer.toString(random.nextInt(1000000000 - 1 + 1) + 1);// generates random number when a new
                                                                                 // user signups
        /*
         * Inserts the information which the users inputs into the form, the SHA2 is a
         * hash and CONCAT will concatenate the salt and the password creating the SHA2
         * hashed password.
         */
        String insert = "INSERT into account" + "(sjsu_id,first_name,last_name,user_name,salt1,password) VALUES "
                + " (?,?,?,?,?,SHA2(CONCAT(?,?),256));";
        int result = 0;

        try (
                // Class.forName("com.mysql.jdbc.Driver");
                Connection connection = DriverManager.getConnection(
                        "jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false", "root",
                        "N00bcakes");

                PreparedStatement preparedStatement = connection.prepareStatement(insert)) {
            preparedStatement.setString(1, acc.getSjsuID());
            preparedStatement.setString(2, acc.getFirstName());
            preparedStatement.setString(3, acc.getLastName());
            preparedStatement.setString(4, acc.getUsername());
            preparedStatement.setString(5, salt1);
            preparedStatement.setString(6, acc.getPassword());
            preparedStatement.setString(7, salt1);

            System.out.println(preparedStatement);
            result = preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return result;

    }
}
```

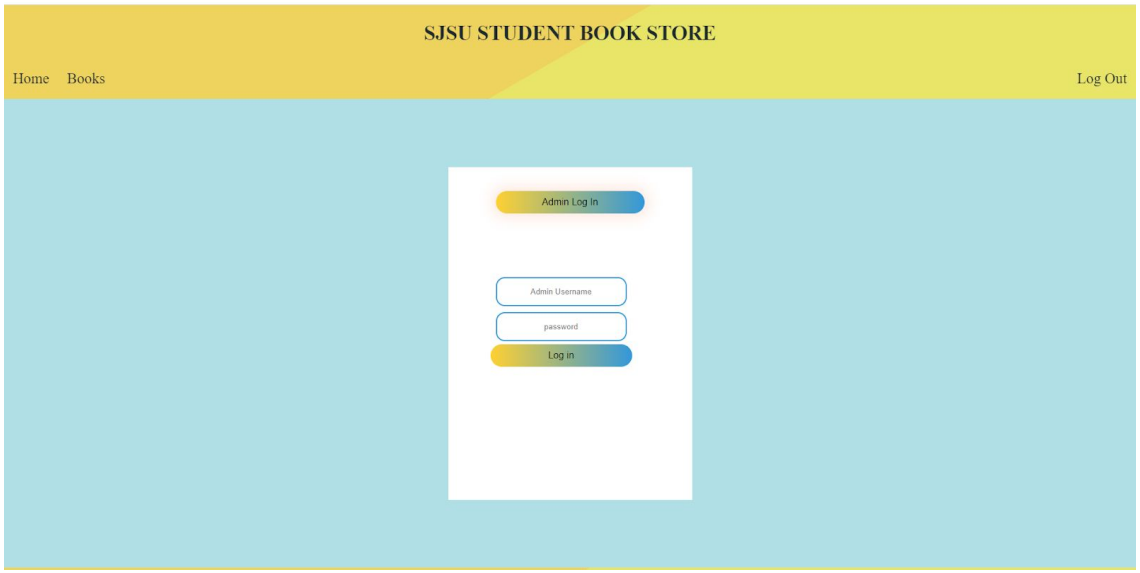
The above screenshot shows the backend code to creating an account. When a user creates an account the data is inserted into the account table.

```
65          <%
66          //Retrieves information from database into table
67      try{
68      Connection connection =
69      DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false","root", "N00bcakes");
70      String account_id=(String)request.getSession().getAttribute("userid");
71      String query = "SELECT sjsu_id, user_name, first_name, last_name from account where account_id = ?";
72
73      PreparedStatement prepared = connection.prepareStatement(query);
74      prepared.setString(1, account_id);
75      ResultSet rs = prepared.executeQuery();
76      rs = prepared.executeQuery();
77      while (rs.next()) {
78          out.println("<tr> <td>" + rs.getString("sjsu_id") + "</td> <td>" +
79          rs.getString("user_name") + "</td><td>" + rs.getString("first_name") + "</td><td>" + rs.getString("last_name") +"</td></tr>");
80    }
81  } catch(SQLException e) {
82      out.println("SQLException caught: " + e.getMessage());
83  }
84
85  %>
```

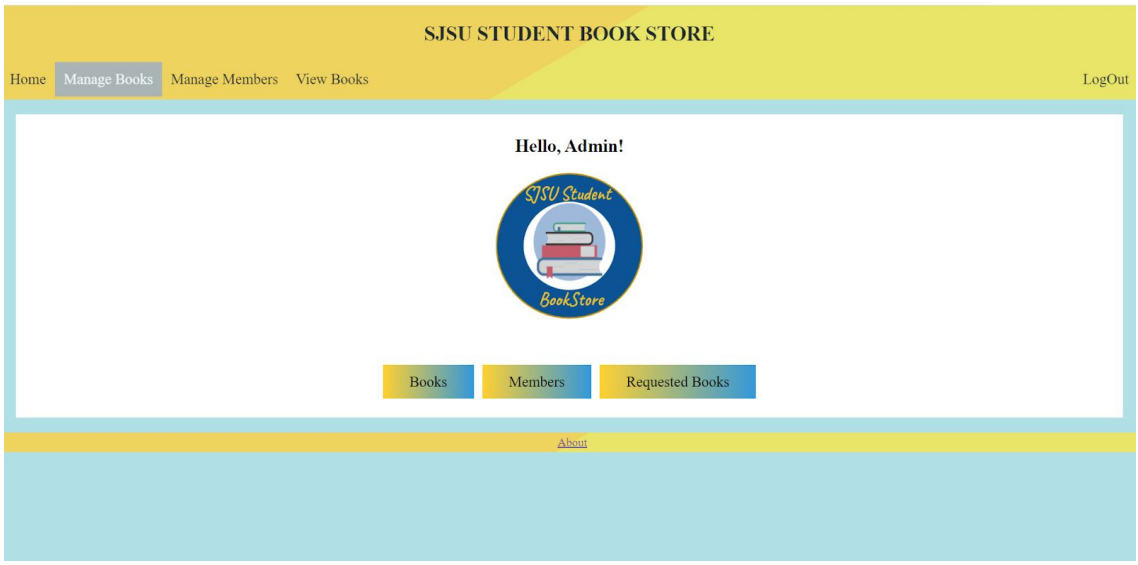
The above screenshot fetches id, first last name, and username from the account table and prints it out in table format. Some html code is embedded in the print statements.

ADMIN

Webpage



The screenshot shows the admin login page where registered admin can login.



The screenshot shows when the admin is logged in, they have various options to choose from. They can manage book, manage members, view book, look at what books they are managing and the members. They can also view the requested book that were made by the members, and add the requested books into the database.

Table

```
mysql> describe admin;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| admin_id  | int         | NO   | PRI | NULL    | auto_increment |
| user_name | varchar(250)| NO   | UNI | NULL    |                |
| salt1     | varchar(250)| NO   |     | NULL    |                |
| password  | varchar(250)| NO   |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.admin;
+----------+-----------+-----------------+----------------------------------------------------------------------+
| admin_id | user_name | salt1           | password                                                             |
+----------+-----------+-----------------+----------------------------------------------------------------------+
|        1 | John      | aadsf43523      | e810f5170bd2fa70cf5ce5f609448ee9ea514270f6150005edef71622a0289ce |
|        2 | Kite      | 313a1dsfa       | e242f043caac8954a3eb6899cf4c145662960198f3d9b8468e6093cf356689eb |
|        3 | Silvya    | 4645afdsf       | c39ebdef4c7eda44e36f8dddab329ae0b757cfcea02c300cfdb0f5f85662ee45 |
|        4 | Blue      | 1324qererasdf   | b2d5429cd2f1686b1f95390c772f1fb009137216350a630dd3106e847c1f5e3c |
|        5 | Green     | asdfasdfa2324   | 0fa21c9f297dc4c4827b06d0df6e0b604edb6e3a384b2138583438790a81b8b1 |
|        6 | Apple     | asdf234         | 4ed36068fc960acfdde41141928c4400633234428e0fb4806f6ca850efa77f5b |
|        7 | August    | 253132a1sdf     | 9efe546b85e349ab809dfa11937d31fd53d576581e8daacac4d03dc5d8649f44 |
|        8 | September | asdf234243      | 2ec6d3a9251059839883a671238f7a19eec403700db8f50427e1ff52bd16228e |
|        9 | Tommy     | asdfafd345345345| a2c208e79c1d503130a4e608a815553ea0c14f4abda2c61cb2e6a0d4881ba6dc |
|       10 | Bob       | 3123213132qer   | 4dd1d6a14582d49a3a668c204a1fe1eceef7eae731357dd5aacaae70104189db |
|       11 | Syam      | asdfa453545435  | fb68222ccaf1f49f03b5faecbea9da3d52e77442eae012b23a05409bfb6d1b91 |
|       12 | Krish     | 654a65sdf16afd  | 56c35e33e1ef308be824974242df3c7946b0ec732b63d615218854d454658028 |
|       13 | Jack      | 54654bjkhabdf   | c37770502587af3aa0a9b2d916f81ed08b47276a98edded7455d8b0a7e72e823 |
|       14 | Daniel    | 45465asdfasdf   | 827383c7d4b4522cadd4c91c1a63082c3207261f8ec24e51402cf0df30aa316f |
|       15 | Regular   | asdrtyryt       | ef236b5d08c735c331e707723b37a10c3c532ba869123f78af0603335e9a7054 |
+----------+-----------+-----------------+----------------------------------------------------------------------+
15 rows in set (0.00 sec)
```

Frontend Code

```
20
21 <!-- Html code that displays the login form of the user -->
22 <body>
23     <div class="sections">
24         <div class="button-box">
25             <div id="btn2"></div>
26             <button type="button" class="toggle-btn" onclick="login()">Admin Log In</button>
27         </div>
28         <!-- DISPLAYs Login FORM -->
29         <form id="login" class="input-group"
30             action="<%=request.getContextPath()%>/Adminlogin" method="post">
31             <input type="text" class="input-field" name="username"
32                 placeholder="Admin Username" required> <input type="password"
33                 class="input-field" name="password" placeholder="password" required>
34             <button type="submit" class="submit-btn" name="adminlogin">Log in</button>
35         </form>
36     </div>
37 </body>
38
```

The above screenshot shows a login form for the admin. Admin must type in username and password and press log in to continue using the web application.

Backend Code

```
93     protected boolean ValidateUser(String username, String password) {
94         String valid_password = null;
95         Connection connection = null;
96         try {
97             try {
98                 connection = DriverManager.getConnection(
99                         "jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false", "root",
100                        "N00bcakes");
101            } catch (Exception e) {
102                System.out.println("Connection Failed: " + e.toString());
103            }
104            PreparedStatement prepared = null;
105            String password_from_db;
106            String salt1;
107            try {
108                prepared = connection.prepareStatement("SELECT * FROM  sjsu_textbookstore.admin WHERE user_name =?");
109                prepared.setString(1, username);
110                ResultSet result = prepared.executeQuery();
111
112                if (result.next()) {
113                    password_from_db = result.getString("password");
114                    salt1 = result.getString("salt1");
115                    // hash user input here with the salt
116                    result.close();
117                    prepared.close();
118                    prepared = connection.prepareStatement(
119                            "Select * from admin where user_name =  ? AND password = sha2(CONCAT( ?, salt1),256);");
120                    prepared.setString(1, username);
121                    prepared.setString(2, password);
122
123                    result = prepared.executeQuery();
124                    if (result.next()) { // user logged in
125                        // User is logged in
126                        return true;
127                    } else {
128                        // not logged in
129                        return false;
130                    }
131                }
132
133            } catch (Exception e) {
134                e.printStackTrace();
135                System.out.println("Cannot validate the account");
```

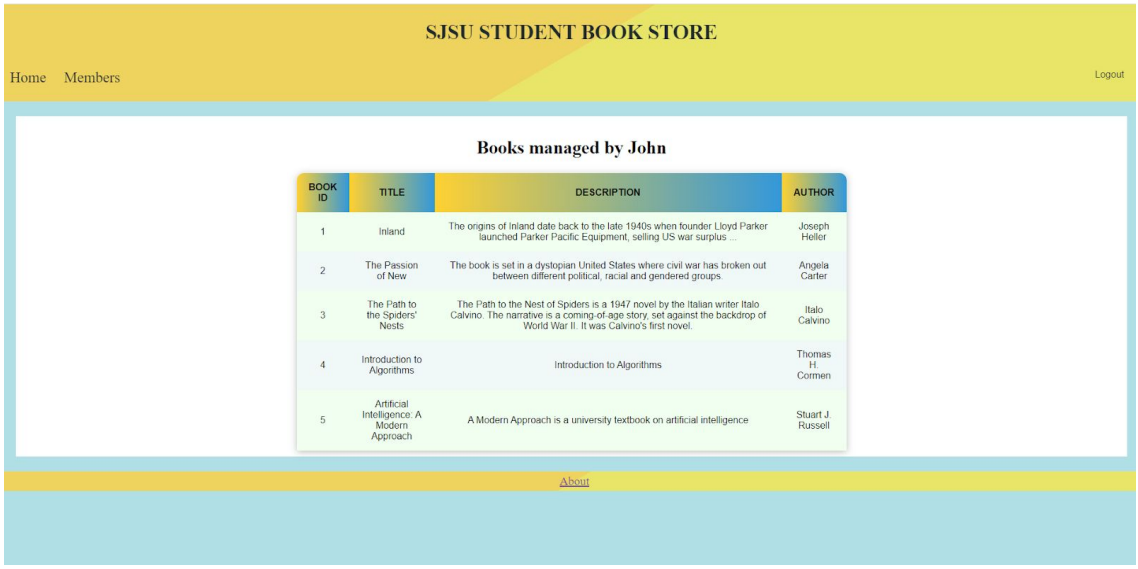
The above screenshot validates the user to see if the person trying to login is actually the admin. The passwords are compared with hashes from the database.

```
47    protected void doPost(HttpServletRequest request, HttpServletResponse response)
48            throws ServletException, IOException {
49        // TODO Auto-generated method stub
50        // doGet(request, response);
51        AdminLoginDAO adminlogindao = new AdminLoginDAO();
52        String username = request.getParameter("username");
53        String password = request.getParameter("password");
54
55        String admin_id = null;
56        try {
57            Connection connection = DriverManager.getConnection(
58                    "jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false", "root",
59                    "N00bcakes");
60
61            // Selecting admin id by selecting the username which the user inputed from the
62            // form
63            String query = "SELECT admin_id From sjsu_textbookstore.admin where user_name = ?";
64            PreparedStatement prepared = connection.prepareStatement(query);
65            prepared.setString(1, username);
66            ResultSet result = prepared.executeQuery();
67            result = prepared.executeQuery();
68            while (result.next()) {
69                admin_id = result.getString("admin_id");
70            }
71            // creating a http session for admin if the username and password was entered
72            // correctly
73            if (ValidateUser(username, password)) {
74                HttpSession session = request.getSession();
75                session.setAttribute("adminid", admin_id);
76                response.sendRedirect("JSP/admin_homepage.jsp");
77            } else {
78                // otherwise it is redirecting it to the login page
79                response.sendRedirect("JSP/login.jsp");
80            }
81
82        } catch (Exception e) {
83            System.out.println("Connection Failed: " + e.toString());
84        }
85    }
```

The screenshot above selects the admin_id from the admin table to create a session. First the username and passwords are retrieved from user input.

ADMIN_BOOKS

Webpage



The above screenshot shows the books that are managed by a specific admin who is logged in, different admin have different books they are assigned to.

Table

```
mysql> describe admin_books;
+----------+------+------+-----+---------+-------+
| Field    | Type | Null | Key | Default | Extra |
+----------+------+------+-----+---------+-------+
| admin_id | int  | NO   | PRI | NULL    |       |
| book_id  | int  | NO   | PRI | NULL    |       |
+----------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.admin_books;
+----------+---------+
| admin_id | book_id |
+----------+---------+
|        1 |       1 |
|        1 |       2 |
|        1 |       3 |
|        1 |       4 |
|        1 |       5 |
|        2 |       6 |
|        2 |       7 |
|        2 |       8 |
|        2 |       9 |
|        2 |      10 |
|        3 |      11 |
|        3 |      12 |
|        3 |      13 |
|        3 |      14 |
|        3 |      15 |
+----------+---------+
15 rows in set (0.00 sec)
```

Frontend code

```
<table class="table-desgin-two">
    <tr>
        <th>Book ID</th>
        <th>Title</th>
        <th>Description</th>
        <th>Author</th>

    </tr>
```

The screenshot above shows the creation of a table with four columns. Table will be used to store data retrieved from admin-books table.

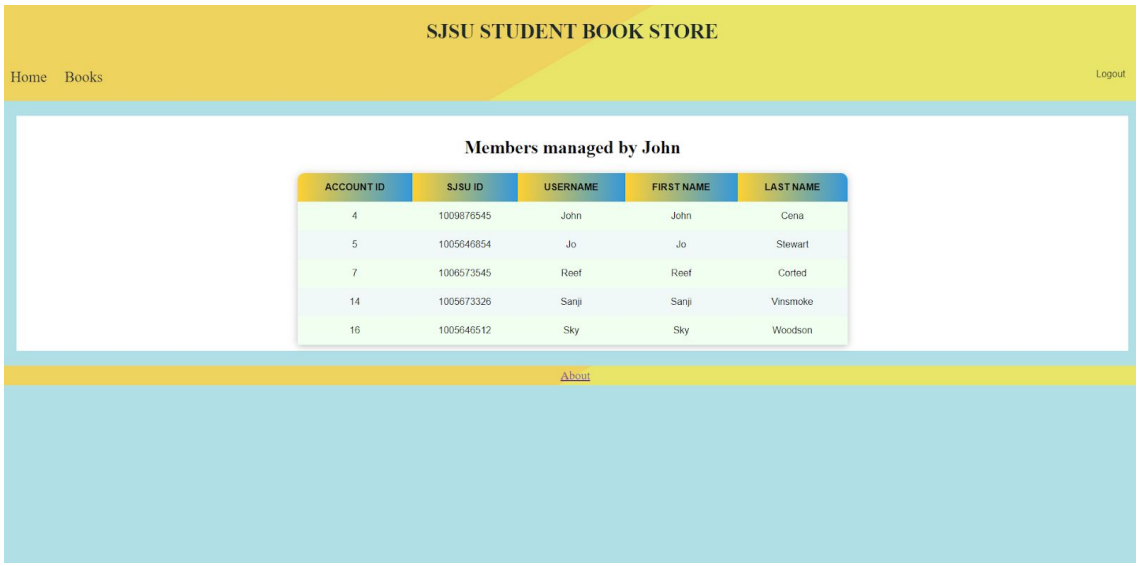
Backend code

```
58
59      <%
60  try{
61      Connection connection =
62      DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false","root", "N00bcakes");
63      String admin_id=(String)request.getSession().getAttribute("adminid");
64      String query = "SELECT * from book,admin_books where admin_id = ? AND admin_books.book_id = book.book_id";
65
66      PreparedStatement prepared = connection.prepareStatement(query);
67      prepared.setString(1, admin_id);
68      ResultSet rs = prepared.executeQuery();
69      rs = prepared.executeQuery();
70      while (rs.next()) {
71          out.println("<tr> <td>" + rs.getString("book_id") + "</td> <td>" +
72          rs.getString("title") + "</td><td>" + rs.getString("description") + "</td><td>" + rs.getString("author") +"</td></tr>");
73    }
74  } catch(SQLException e) {
75      out.println("SQLException caught: " + e.getMessage());
76  }
77
78  %>
```

The screenshot above shows a query which selects from admin_books and book that match id. The results of the books that are managed by this admin will be displayed on a table. Html formatting is embedded in the print statements.

ADMIN_Members

Webpage



The above screenshot shows the members that are managed by a specific admin who is logged in, different admin have different books they are assigned to.

Table

```
mysql> describe admin_members;
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| admin_id   | int  | NO   | PRI | NULL    |                |
| account_id | int  | NO   | PRI | NULL    | auto_increment |
+------------+------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.admin_members;
+----------+------------+
| admin_id | account_id |
+----------+------------+
|        3 |          1 |
|        2 |          2 |
|        3 |          3 |
|        1 |          4 |
|        1 |          5 |
|        2 |          6 |
|        1 |          7 |
|        2 |          8 |
|        3 |          9 |
|        3 |         10 |
|        3 |         11 |
|        3 |         12 |
|        2 |         13 |
|        1 |         14 |
|        3 |         15 |
|        1 |         16 |
+----------+------------+
16 rows in set (0.00 sec)
```

Frontend Code

```
<!-- Table that displays all the members information  -->
<table class="table-desgin-two">
    <tr>
        <th>Account ID</th>
        <th>SJSU ID</th>
        <th>Username</th>
        <th>First Name</th>
        <th>Last Name</th>
    </tr>
```

This screenshot aboves shows a creation of table that displays all the members information. The data and html is embedded in the backend code.

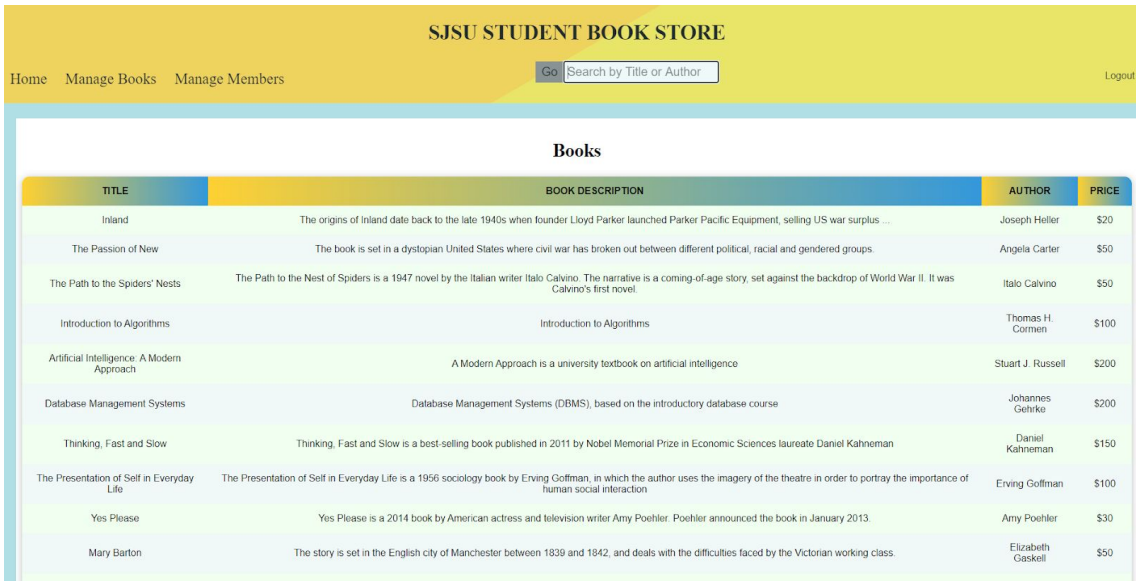
Backend Code

```
<% /**Establish connection of database and then retrieve information about members that are mananged by
    this admin.  Displays sjsu id, first name, and last name.
**/
try{
    Connection connection =
    DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false","root", "N00bcakes")
    String admin_id=(String)request.getSession().getAttribute("adminid");
    String query = "SELECT account.account_id,sjsu_id, account.user_name, first_name, last_name"
    +" FROM account,admin_members where admin_id = ? AND admin_members.account_id = account.account_id";
    PreparedStatement prepared = connection.prepareStatement(query);
    prepared.setString(1, admin_id);
    ResultSet rs = prepared.executeQuery();
    rs = prepared.executeQuery();
    while (rs.next()) {
        out.println("<tr><td>" +rs.getString("account_id")+"</td><td>" + rs.getString("sjsu_id") + "</td> <td>" +
        rs.getString("user_name") + "</td><td>" + rs.getString("first_name") + "</td><td>" + rs.getString("last_name") +"</td></tr>")
    }
} catch(SQLException e) {
    out.println("SQLException caught: " + e.getMessage());
}
%>
</table>
```

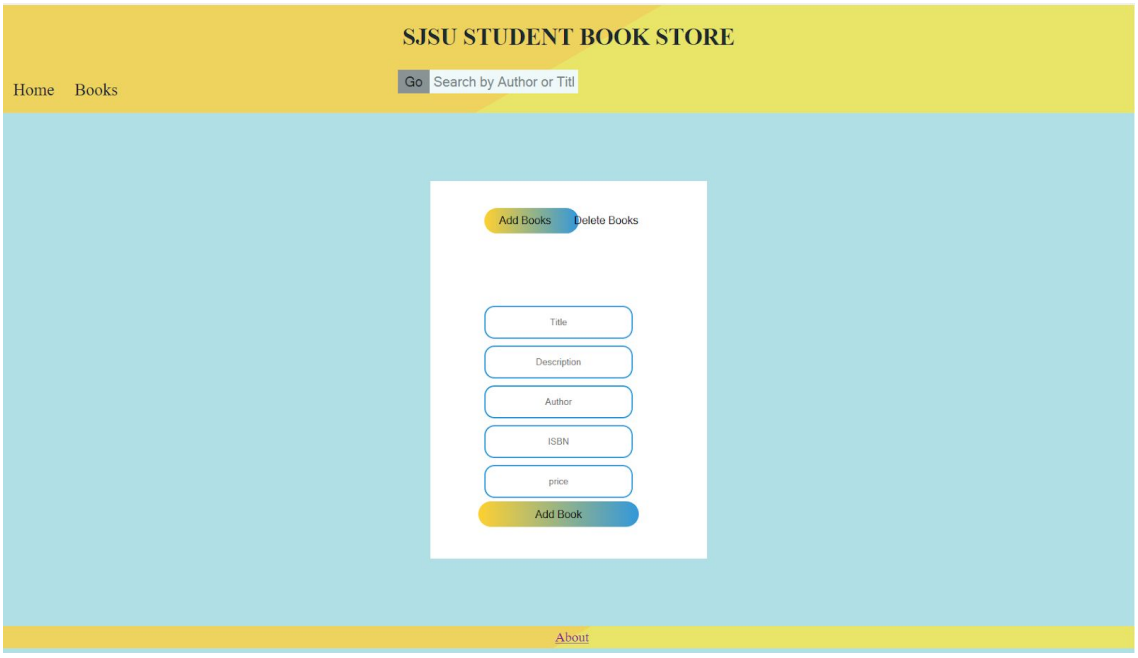
This screenshot above retrieves the members that are managed by this data using a select statement. The while loop prints the results using print statements with html code embedded inside.

Book

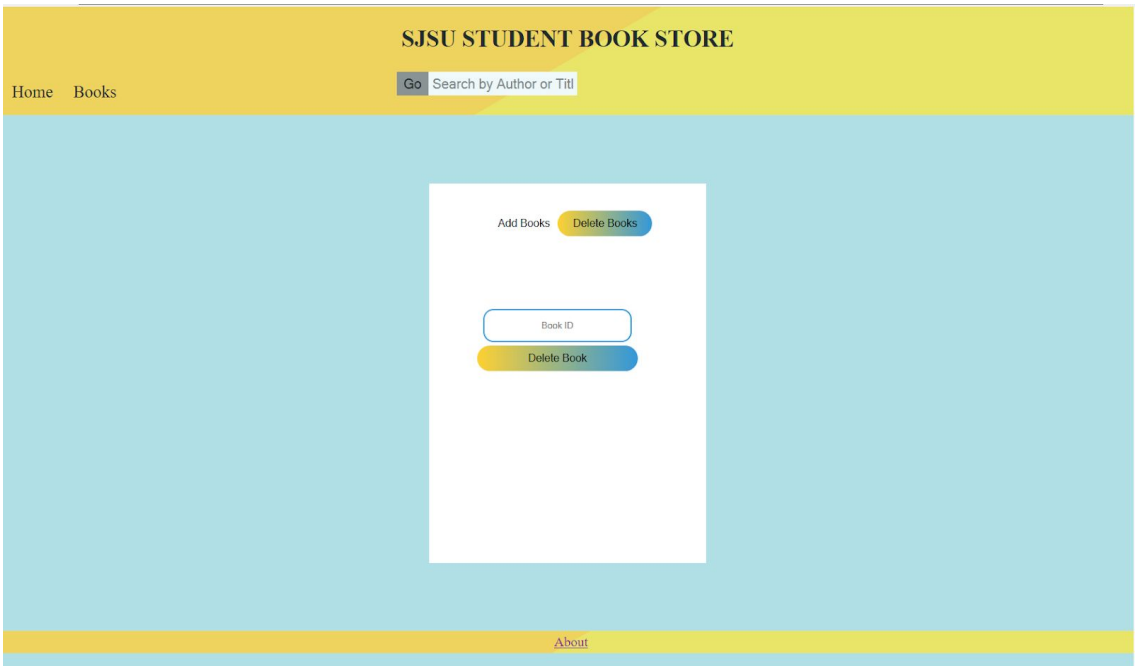
Webpage



The screenshot shows a search bar, where admin or members can search for books by entering the book title or the author name.

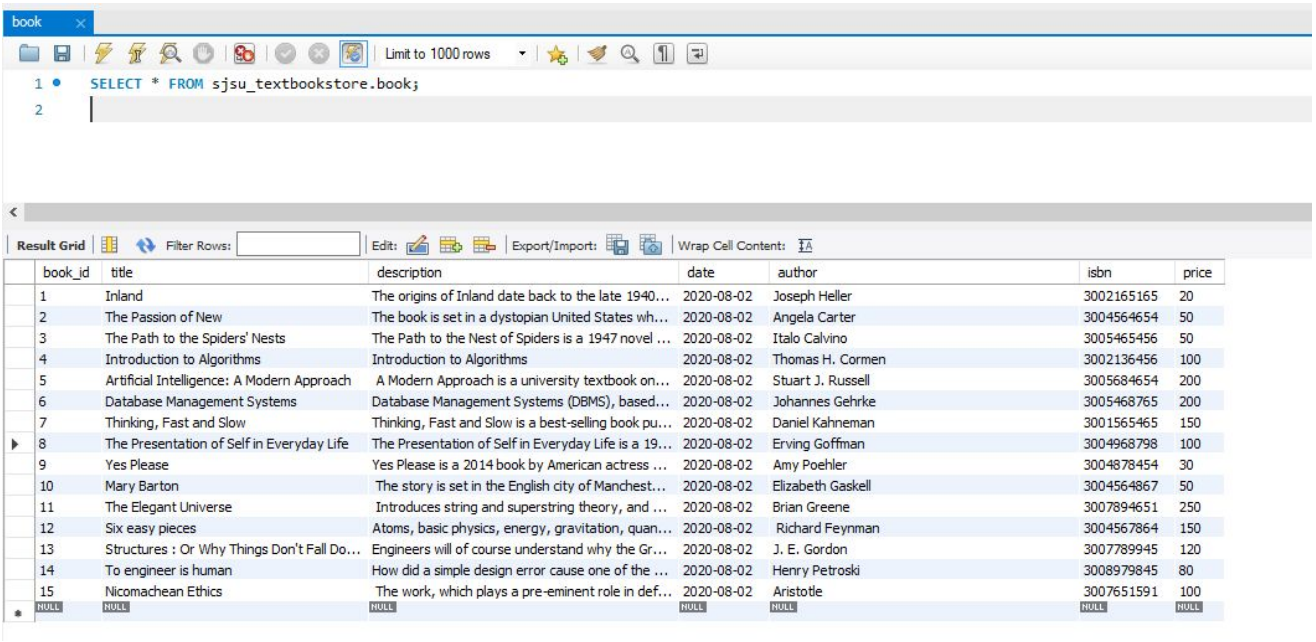


The above screenshot shows the adding books, where the admin united all the form information to add the book.



The above screenshot shows a deleting book where the admin enters the book Id to delete a book from the database.

Table



book_id	title	description	date	author	isbn	price
1	Inland	The origins of Inland date back to the late 1940...	2020-08-02	Joseph Heller	3002165165	20
2	The Passion of New	The book is set in a dystopian United States wh...	2020-08-02	Angela Carter	3004564654	50
3	The Path to the Spiders' Nests	The Path to the Nest of Spiders is a 1947 novel ...	2020-08-02	Italo Calvino	3005465456	50
4	Introduction to Algorithms	Introduction to Algorithms	2020-08-02	Thomas H. Cormen	3002136456	100
5	Artificial Intelligence: A Modern Approach	A Modern Approach is a university textbook on...	2020-08-02	Stuart J. Russell	3005684654	200
6	Database Management Systems	Database Management Systems (DBMS), based...	2020-08-02	Johannes Gehrke	3005468765	200
7	Thinking, Fast and Slow	Thinking, Fast and Slow is a best-selling book pu...	2020-08-02	Daniel Kahneman	3001565465	150
8	The Presentation of Self in Everyday Life	The Presentation of Self in Everyday Life is a 19...	2020-08-02	Erving Goffman	3004968798	100
9	Yes Please	Yes Please is a 2014 book by American actress ...	2020-08-02	Amy Poehler	3004878454	30
10	Mary Barton	The story is set in the English city of Manchest...	2020-08-02	Elizabeth Gaskell	3004564867	50
11	The Elegant Universe	Introduces string and superstring theory, and ...	2020-08-02	Brian Greene	3007894651	250
12	Six easy pieces	Atoms, basic physics, energy, gravitation, quan...	2020-08-02	Richard Feynman	3004567864	150
13	Structures : Or Why Things Don't Fall Do...	Engineers will of course understand why the Gr...	2020-08-02	J. E. Gordon	3007789945	120
14	To engineer is human	How did a simple design error cause one of the ...	2020-08-02	Henry Petroski	3008979845	80
15	Nicomachean Ethics	The work, which plays a pre-eminent role in def...	2020-08-02	Aristotle	3007651591	100
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Frontend Code

```
<form action="books_display.jsp" autocomplete="off">
    <input type="text" placeholder="Search by Title or Author"
        name="search">
    <button type="submit">
        <i class="fa fa-search"></i>Go
    </button>
</form>
```

The screenshot aboves a form where users can use search books by title or author.

```
<!-- Table that displays the books result based on user preference -->
<table class="table-desgin">
    <tr>
        <th>Title</th>
        <th>Book Description</th>
        <th>Author</th>
        <th>Price</th>
    </tr>
```

The screenshot above shows the creation of a table with four columns that displays the information on the book. There were multiple instances where this same code was used throughout the application. Html is also embedded in the backend code.

```
<!-- Admin has option to add or delete books from the database system. Admin must fill out
 a form of the book they want to add or delete. -->
<body>
    <div class="sections">
        <div class="button-box">
            <div id="btn"></div>
            <button type="button" class="toggle-btn" onclick="addbook()">Add Books</button>
            <button type="button" class="toggle-btn" onclick="deletebook()">Delete Books</button>
        </div>
        <!-- DISPLAY Login FORM -->
        <form id="addbooks" class="input-group"
            action="<%=request.getContextPath()%>/addbooks" method="post">
            <input type="text" class="input-field" name="title"
                placeholder="Title" required>

            <input type="text" class="input-field" name="description"
                placeholder="Description" required>
            <input type="text" class="input-field" name="author"
                placeholder="Author" required>
            <input type="text" class="input-field" name="isbn"
                placeholder="ISBN" required>
            <input type="text" class="input-field" name="price"
                placeholder="price" required>
            <button type="submit" class="submit-btn" name="addbooks">Add Book</button>
        </form>

        <form id="deletebooks" class="input-group"
            action="<%=request.getContextPath()%>/deletebooks" method="post">
            <input type="text" class="input-field" name="book_id"
                placeholder="Book ID" required>
            <button type="submit" class="submit-btn" name="deletebooks" value="deletebooks" >Delete Book</button>
        </form>
        <br>
    </div>
```

This above screenshot shows the Add and Delete book form for admin. Admin enters information in the database and can either add or delete a book.

Backend Code

```
public class Book {
    private String book_id;
    private String title;
    private String description;
    private String Author;
    private String price;
    private String ISBN;

    public Book(String t, String d, String a, String p, String isbn) {

        this.title = t;
        this.description = d;
        this.Author = a;
        this.price = p;
        this.ISBN = isbn;

    }

    public Book(String book_id) {

        this.book_id = book_id;

    }

    public String getBook_id() {
        return book_id;
    }

    public void setBook_id(String book_id) {
        this.book_id = book_id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
```

The above screenshot is the book class with setters and getters.

```
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getAuthor() {
        return Author;
    }

    public void setAuthor(String author) {
        Author = author;
    }

    public String getPrice() {
        return price;
    }

    public void setPrice(String price) {
        this.price = price;
    }

    public String getIsbn() {
        return ISBN;
    }

    public void setISBN(String isbn) {
        this.ISBN = isbn;
    }

}
```

The above screenshot is a continuation of the previous screenshot containing the book class with setters and getters.

```
public class AdminDeleteBooksDAO {
    public int deleteBook(Book deleteBook) throws Exception {
        /*
         * This will delete the book using the book Id
         */
        String insert = "DELETE from book where book_id = ?";
        int result = 0;

        try (
                // Class.forName("com.mysql.jdbc.Driver");
                Connection connection = DriverManager.getConnection(
                        "jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false", "root",
                        "N00bcakes");

                PreparedStatement preparedStatement = connection.prepareStatement(insert)) {
            preparedStatement.setString(1, (deleteBook.getBook_id()));

            System.out.println(preparedStatement);

            result = preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return 0;

    }
}
```

The above screenshot deletes the book from the database after the admin fills out the form to delete the book.

```
public int addBook(Book addBook) throws Exception {

    /*
     * This is to add the books which the admin enters into the form, the query is
     * for inserting books at the book table.
     */

    String insert = "INSERT into book" + "(title,description,date,author,isbn,price) VALUES " + " (?,?,?,?,?,?)";
    int result = 0;

    try (
            // Class.forName("com.mysql.jdbc.Driver");
            Connection connection = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false", "root",
                    "N00bcakes");

            PreparedStatement preparedStatement = connection.prepareStatement(insert)) {
        preparedStatement.setString(1, addBook.getTitle());
        preparedStatement.setInt(2, acc.getAccount_id());
        preparedStatement.setString(2, addBook.getDescription());
        preparedStatement.setDate(3, getCurrentDate());
        preparedStatement.setString(4, addBook.getAuthor());
        preparedStatement.setString(5, addBook.getIsbn());
        preparedStatement.setString(6, addBook.getPrice());

        System.out.println(preparedStatement);
        result = preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return 0;

}
```

The above screenshot inserts the book to the database after the admin fills out the form to add the book.

```
<%
String userInput = request.getParameter("search");
String db = "sjsu_textbookstore";
String user;
user = "root";
String password = "N00bcakes";
try {
    java.sql.Connection con;
    Class.forName("com.mysql.jdbc.Driver");
    con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false",
    user, password);
    Statement stmt = con.createStatement();
    ResultSet rs = stmt
    .executeQuery("SELECT * FROM book WHERE title = '" + userInput + "' OR author = '" + userInput + "'");
    if (userInput != null) {
        boolean empty = rs.next();
        if (empty == false) {
    out.println("<b>No results from your search were found <br> Try checking our list of books</b> <br>");
    rs = stmt.executeQuery("SELECT * FROM book");
    while (rs.next()) {
        out.println("<tr> <td>" + rs.getString("title") + "</td> <td>" + rs.getString("description")
                + "</td><td>" + rs.getString("author") + "</td><td> $" + rs.getString("price") + "</td></tr>");}
        } else {
    String account_id = (String) request.getSession().getAttribute("userid");
    String query = "INSERT INTO recent_search (book_id,account_id) VALUES ('" + rs.getInt("book_id") + "', '"
            + account_id + "')";
    PreparedStatement prepared = con.prepareStatement(query);
    prepared.execute();
    rs = stmt.executeQuery("SELECT * FROM book WHERE title = '" + userInput + "' OR author = '" + userInput + "'");
    while (rs.next()) {
        out.println("<tr> <td>" + rs.getString("title") + "</td> <td>" + rs.getString("description")
                + "</td><td>" + rs.getString("author") + "</td><td> $" + rs.getString("price") + "</td></tr>");} }
        rs.close();
        stmt.close();
        con.close();
    } else {
        rs = stmt.executeQuery("SELECT * FROM book");
        while (rs.next()) {
    out.println("<tr> <td>" + rs.getString("title") + "</td> <td>" + rs.getString("description") + "</td><td>"
            + rs.getString("author") + "</td><td> $" + rs.getString("price") + "</td></tr>");}}
} catch (SQLException e) {
    out.println("SQLException caught: " + e.getMessage());}
%>
```

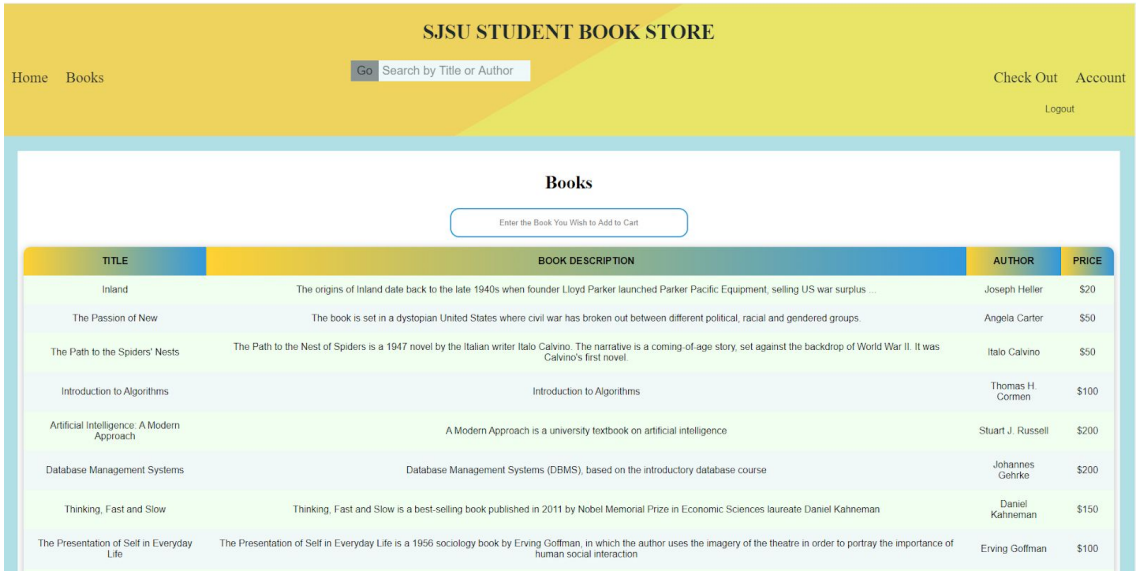
The screenshot above selects all the books from the search bar used by the user. Initially when the page loads, there is no query, so all the books are displayed. If a user uses the search query, then the results will be returned based on user searches. If a search is invalid, the system prompts a statement saying invalid and all the books are displayed again as suggestions.

```
47
48      <% //Selects the books that match user input and inserts it into the checkout cart
49      String addingBooks = request.getParameter("booksToAdd");
50      String database = "sjsu_textbookstore";
51      String users;
52      users = "root";
53      String passwords = "N00bcakes";
54      try {
55          java.sql.Connection con;
56          Class.forName("com.mysql.jdbc.Driver");
57          con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false",
58          users, passwords);
59          //Create sql statement
60          Statement stmt = con.createStatement();
61          if (addingBooks != null) {
62              ResultSet rs = stmt.executeQuery("SELECT * FROM book WHERE title = '" + addingBooks + "'");
63              boolean empty = rs.next();
64              //If no results are found then display message
65              if (empty == false) {
66          out.println("<b>Please enter a valid book</b><br>");
67              }
68              //Display the results from the query given by the user, the book is stored in the database for recently searched
69              else {
70          String account_id = (String) request.getSession().getAttribute("userid");
71          String query = "INSERT INTO checkout(book_id, account_id, title) VALUES ('" + rs.getString("book_id")
72                  + "', '" + account_id + "', '" + rs.getString("title") + "')";
73          PreparedStatement prepared = con.prepareStatement(query);
74          prepared.execute();
75              }
76              rs.close();
77              stmt.close();
78              con.close();
79          }
80      } catch (SQLException e) {
81          out.println("SQLException caught: " + e.getMessage());
82      }
83      %>
```

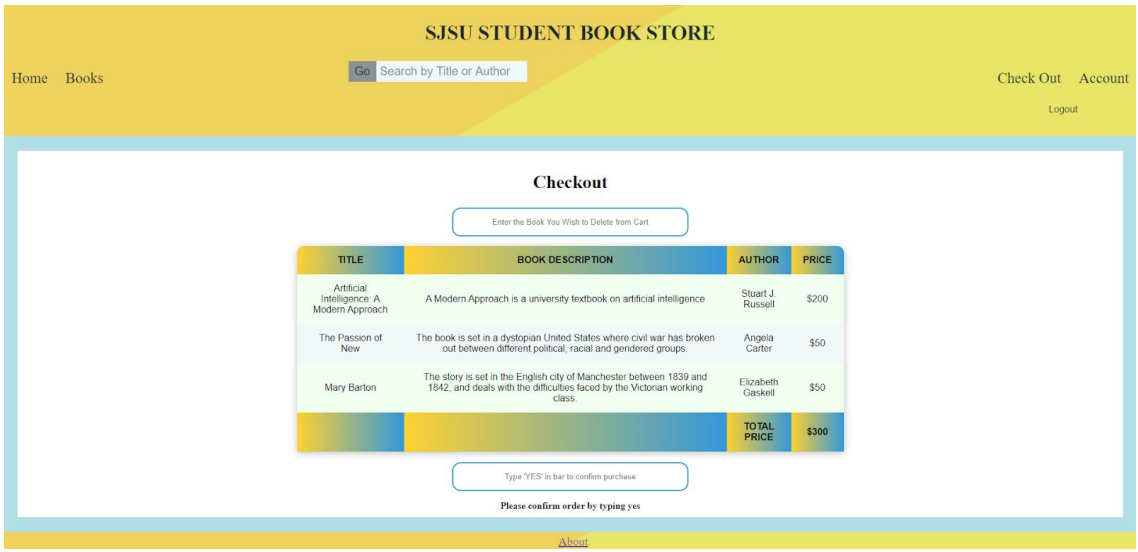
The screenshot above selects books that match the user input. The search result is used to add a book into the checkout table.

Checkout

Webpage



The above screenshot shows an add to cart text field. Right underneath the Books header, the text field is used for members to enter the books they wish to add into their checkout later to buy it.



The above screenshot shows the checkout page of the users who are logged in. It shows their cart containing all the books they added. Now when they want to delete the book

from the checkout, they can enter the book name on the text field below the Checkout header.

Table

```
mysql> describe checkout;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| checkout_id | int          | NO   | PRI | NULL    | auto_increment |
| book_id     | int          | NO   | PRI | NULL    |                |
| account_id  | int          | NO   | PRI | NULL    |                |
| title       | varchar(250) | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.checkout;
+-------------+---------+------------+-----------------------------------------+
| checkout_id | book_id | account_id | title                                   |
+-------------+---------+------------+-----------------------------------------+
|          20 |       5 |         14 | Artificial Intelligence: A Modern Approach |
|          21 |       2 |         14 | The Passion of New                      |
|          22 |      10 |         14 | Mary Barton                             |
|          26 |      11 |          9 | The Elegant Universe                    |
|          27 |      15 |          9 | Nicomachean Ethics                      |
|          30 |      12 |         11 | Six easy pieces                         |
|          31 |       8 |         11 | The Presentation of Self in Everyday Life |
|          32 |       9 |         11 | Yes Please                              |
|          36 |       8 |          5 | The Presentation of Self in Everyday Life |
|          37 |       1 |          5 | Inland                                  |
|          38 |       9 |          5 | Yes Please                              |
|          39 |       7 |          5 | Thinking, Fast and Slow                 |
|          40 |       2 |          5 | The Passion of New                      |
|          48 |       6 |          7 | Database Management Systems             |
|          49 |       8 |          7 | The Presentation of Self in Everyday Life |
+-------------+---------+------------+-----------------------------------------+
15 rows in set (0.00 sec)
```

Frontend Code

```
40
41          <!-- Form for users to add boooks to their checkout cart -->
42          <form action="./books_display.jsp" method="POST" class="input-group-1">
43              <input class="input-field-1" type="text"
44                  placeholder="Enter the Book You Wish to Add to Cart"
45                  name="booksToAdd" />
46          </form>
```

This screenshot displays the form which asks for user input on the book they want to add.

```
82
83          <!-- Displays the table of books that are in the user checkout cart -->
84          <table class=table-desgin-two>
85              <tr>
86                  <th>Title</th>
87                  <th>Book Description</th>
88                  <th>Author</th>
89                  <th>Price</th>
90              </tr>
```

This screenshot displays the creation of the checkout table with column names. The data display is embedded inside the backend code.

Backend Code

```
47
48      <% //Selects the books that match user input and inserts it into the checkout cart
49      String addingBooks = request.getParameter("booksToAdd");
50      String database = "sjsu_textbookstore";
51      String users;
52      users = "root";
53      String passwords = "N00bcakes";
54      try {
55          java.sql.Connection con;
56          Class.forName("com.mysql.jdbc.Driver");
57          con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false",
58          users, passwords);
59          //Create sql statement
60          Statement stmt = con.createStatement();
61          if (addingBooks != null) {
62              ResultSet rs = stmt.executeQuery("SELECT * FROM book WHERE title = '" + addingBooks + "'");
63              boolean empty = rs.next();
64              //If no results are found then display message
65              if (empty == false) {
66          out.println("<b>Please enter a valid book</b><br>");
67              }
68              //Display the results from the query given by the user, the book is stored in the database for recently searched
69              else {
70          String account_id = (String) request.getSession().getAttribute("userid");
71          String query = "INSERT INTO checkout(book_id, account_id, title) VALUES ('" + rs.getString("book_id")
72                  + "', '" + account_id + "', '" + rs.getString("title") + "')";
73          PreparedStatement prepared = con.prepareStatement(query);
74          prepared.execute();
75              }
76              rs.close();
77              stmt.close();
78              con.close();
79          }
80      } catch (SQLException e) {
81          out.println("SQLException caught: " + e.getMessage());
82      }
83      %>
```

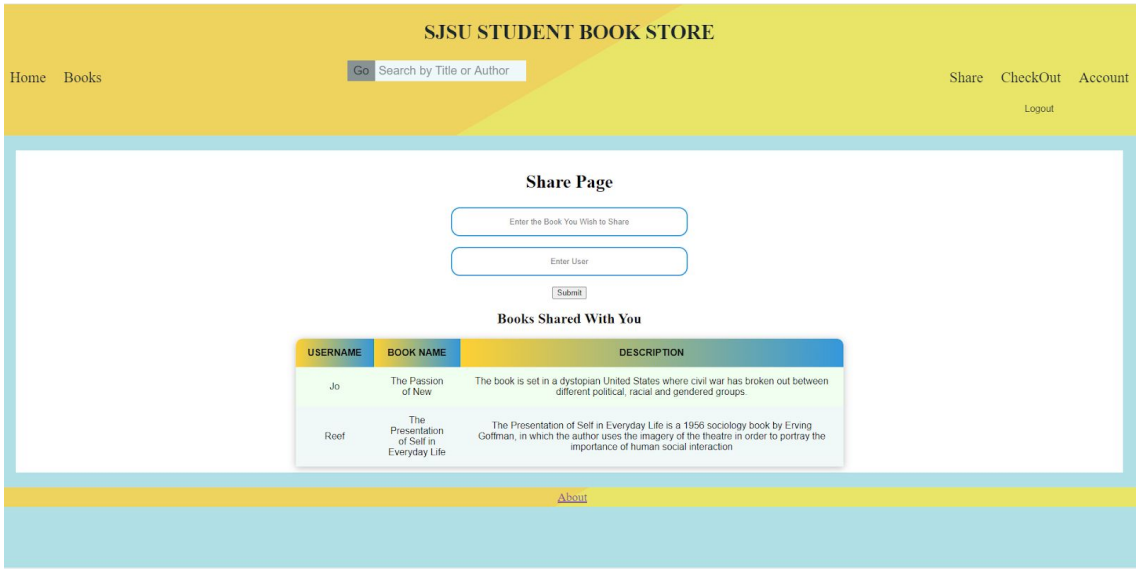
This screenshot shows how to inserts book into the checkout table database.

```
91
92          <% //Selects the books that user has added to cart
93          int sum = 0;
94          String db = "sjsu_textbookstore";
95          String user;
96          user = "root";
97          String password = "N00bcakes";
98          try {
99              java.sql.Connection con;
100             Class.forName("com.mysql.jdbc.Driver");
101             con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?serverTimezone=EST5EDT", user,
102             password);
103             Statement stmt = con.createStatement();
104             String account_id = (String) request.getSession().getAttribute("userid");
105             ResultSet rs = stmt.executeQuery(
106             "SELECT * FROM book, checkout WHERE account_id = '"+account_id+"' AND book.book_id = checkout.book_id");
107             boolean empty = rs.next();
108             if (empty == false) {
109                 out.println("<b>Your shopping cart is empty <br> Please add items you wish to purchase</b> <br>");
110             } else {
111
112                 do {
113             out.println("<tr><td>" + rs.getString("title") + "</td> <td>" + rs.getString("description") + "</td><td>"
114                     + rs.getString("author") + "</td><td> $" + rs.getString("price") + "</td></tr>");
115             sum = sum + rs.getInt("price");
116                 } while (rs.next());
117             }
118             out.println("<tr><th></th><th></th><th>Total price</th><th> $" + sum + "</th></tr>");
119             rs.close();
120             stmt.close();
121             con.close();
122         } catch (SQLException e) {
123             out.println("SQLException caught: " + e.getMessage());
124         }
125         %>
126
```

This screenshot shows how to select book from checkout table and display data as a table.

Member Share

Webpage



The above screenshot shows the member share page, where members can share the books with other members. Shared books are shown via tables.

Table

```
mysql> describe member_share;
+-------------+------+------+-----+---------+----------------+
| Field       | Type | Null | Key | Default | Extra          |
+-------------+------+------+-----+---------+----------------+
| share_id    | int  | NO   | PRI | NULL    | auto_increment |
| sender_id   | int  | NO   |     | NULL    |                |
| book_id     | int  | NO   | PRI | NULL    |                |
| receiver_id | int  | NO   |     | NULL    |                |
+-------------+------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.member_share;
+----------+-----------+---------+-------------+
| share_id | sender_id | book_id | receiver_id |
+----------+-----------+---------+-------------+
|        1 |         1 |       1 |           1 |
|        2 |         3 |      11 |          13 |
|        3 |         3 |      11 |          13 |
|        4 |        14 |       7 |           3 |
|        5 |        11 |       8 |           1 |
|        6 |        11 |       4 |           3 |
|        7 |        11 |      12 |           5 |
|        8 |         5 |       2 |          14 |
|        9 |         5 |      12 |           2 |
|       10 |         5 |      11 |          13 |
|       11 |         7 |       8 |           5 |
|       12 |         7 |       8 |           1 |
|       13 |         7 |       8 |           2 |
|       14 |         7 |       8 |          14 |
|       15 |         7 |       8 |           8 |
+----------+-----------+---------+-------------+
15 rows in set (0.00 sec)
```

Frontend Code

```
34 <body>
35     <section>
36     <h1>Share Page</h1>
37         <!-- Form where users can enter the book that they want to share with another user -->
38         <form  action = "./share_book.jsp" method = "POST" class="input-group-1" >
39             <input class = "input-field-1" type= "text" placeholder="Enter the Book You Wish to Share" name= "booksShare" />
40             <br><br>
41             <input class = "input-field-1" type= "text" placeholder="Enter User" name= "usersShare" />
42             <br>
43             <br>
44             <input type = "submit" >
45         </form>
```

The screenshot above will display a form that asks user to enter the book they want to share and to which user.

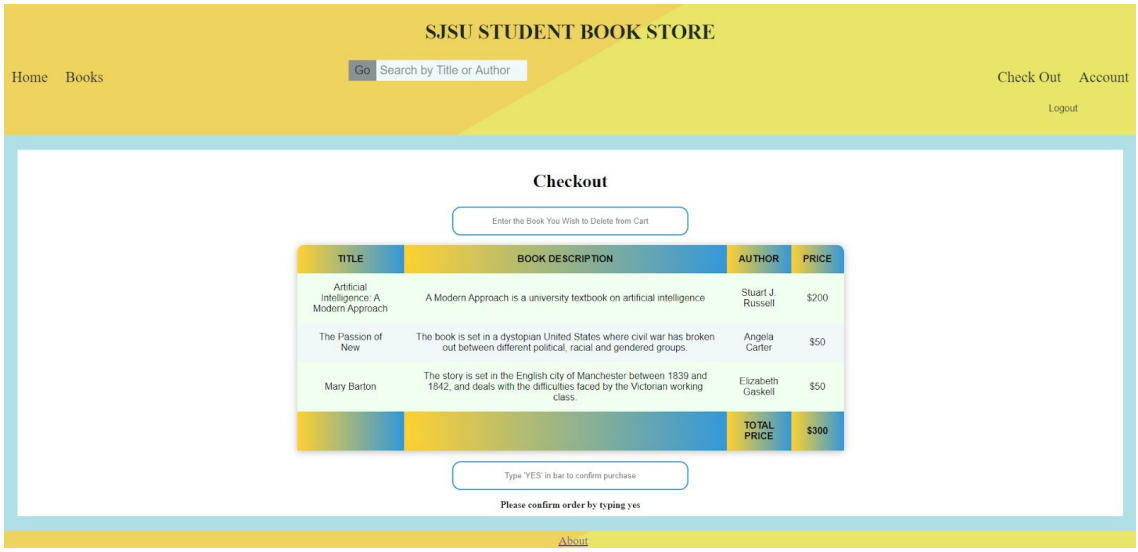
Backend Code

```
47        <% //Takes in the input that was entered by the user with book share and user share
48  try{
49      String booksShare = request.getParameter("booksShare");
50      String usersShare  = request.getParameter("usersShare");
51      Connection connection =
52      DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false","root", "N00bcakes");
53      Statement stmt = connection.createStatement();
54      Statement stmt1 = connection.createStatement();
55      if(booksShare != null && usersShare != null)
56      {
57
58          ResultSet rs1 = stmt.executeQuery("SELECT book_id FROM book WHERE title = '"+booksShare+"'");
59          ResultSet rs2 = stmt1.executeQuery("SELECT account_id FROM account WHERE user_name = '"+usersShare+"'");
60          rs1.next();
61          rs2.next();
62          String accounts_id=(String)request.getSession().getAttribute("userid");
63          int accountss_id = Integer.parseInt(accounts_id);
64          String query =
65          "INSERT INTO member_share(sender_id, book_id, receiver_id) VALUES" +
66          " ('"+accountss_id+"', '"+rs1.getInt("book_id")+"', '"+rs2.getInt("account_id")+"')";
67          PreparedStatement prepared = connection.prepareStatement(query);
68          prepared.execute();
69       }
70
71  } catch(SQLException e) {
72      out.println("SQLException caught: " + e.getMessage());
73  }
74      %>
```

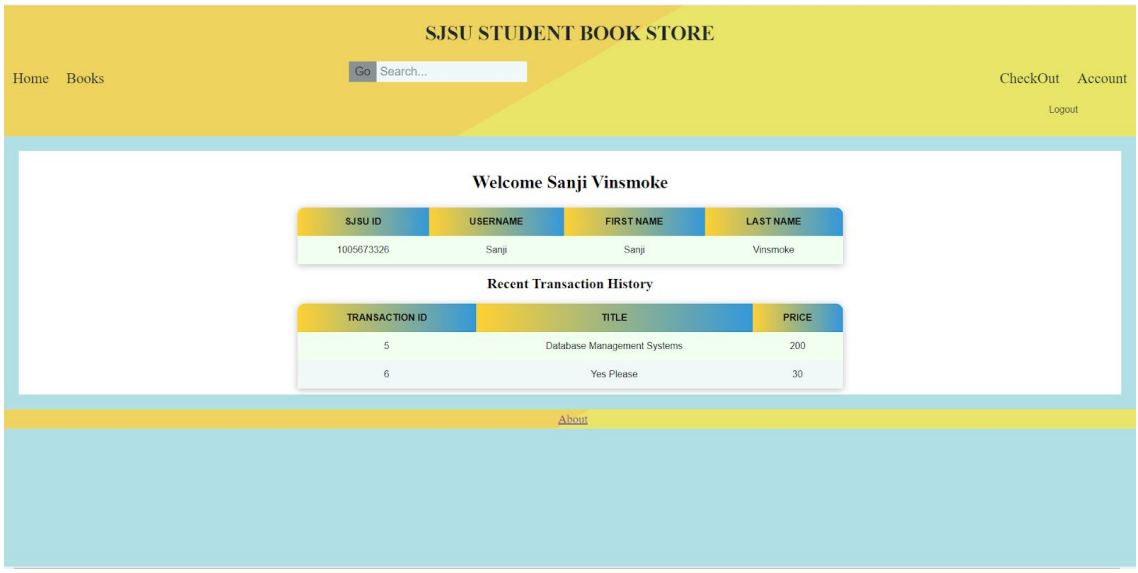
The screenshot above takes in the inputs by a user and searches the database for results. If both forms returns a valid results then the retrieved data will be inserted into member_share table.

Purchase

Webpage



The screenshot above shows the checkout page, where the user can purchase the book from their checkout cart. In order for the user to purchase the book, they must confirm it by inputting yes, in the bottom text field.



The screenshot above shows the purchase history of the member who has bought books and is logged in.

Table

```
mysql> describe purchase;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| transaction_id | int          | NO   | PRI | NULL    | auto_increment |
| account_id     | int          | NO   | PRI | NULL    |                |
| book_id        | int          | NO   | PRI | NULL    |                |
| price          | varchar(250) | NO   |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.purchase;
+----------------+------------+---------+-------+
| transaction_id | account_id | book_id | price |
+----------------+------------+---------+-------+
|              1 |          1 |      10 | 50    |
|              2 |          1 |      12 | 150   |
|              3 |          2 |       2 | 50    |
|              4 |          3 |       2 | 50    |
|              5 |         14 |       6 | 200   |
|              6 |         14 |       9 | 30    |
|              7 |          9 |      11 | 250   |
|              8 |          9 |       9 | 30    |
|              9 |         11 |       8 | 100   |
|             10 |          5 |      10 | 50    |
|             11 |          5 |      15 | 100   |
|             12 |          7 |       4 | 100   |
|             13 |          7 |       8 | 100   |
|             14 |          7 |      15 | 100   |
|             15 |          7 |       2 | 50    |
+----------------+------------+---------+-------+
15 rows in set (0.00 sec)
```

Frontend Code

```
<!-- Form for user to confirm their purchase -->
<form action="./checkout.jsp" method="POST" class="input-group-1">
    <input class="input-field-1" type="text"
        placeholder="Type 'YES' in bar to confirm purchase"
        name="confirmation" />
</form>
```

This screenshot above will display a form that asks the user to confirm their purchase.

```
<!-- Recent transaction table display with information retrieved from database  -->
<h2>Recent Transaction History</h2>
<table class="table-desgin-two">
    <tr>
        <th>Transaction ID</th>
        <th>Title</th>
        <th>Price</th>
    </tr>
```

The screenshot above creates a table with three column names. This html will also be embedded into the backend code.

Backend Code

```
if (yes != null && (yes.equals("yes") || yes.equals("YES") || yes.equals("Yes") || yes.equals("YEs")|| yes.equals("YeS"))) {
    PreparedStatement prepared = con.prepareStatement(query);
    prepared.setString(1, account_id);
    ResultSet rs = prepared.executeQuery();
    rs = prepared.executeQuery();
    boolean empty = rs.next();
    //If no results are found then display message
    if (empty != false) {
        do {
            String query1 = "INSERT INTO purchase (account_id,book_id, price )" +
                    "VALUES ('"+account_id+"', '" + rs.getInt("book_id") + "','" + rs.getInt("price") + "')";
            PreparedStatement prepared1 = con.prepareStatement(query1);
            prepared1.execute();
        }while (rs.next());
```

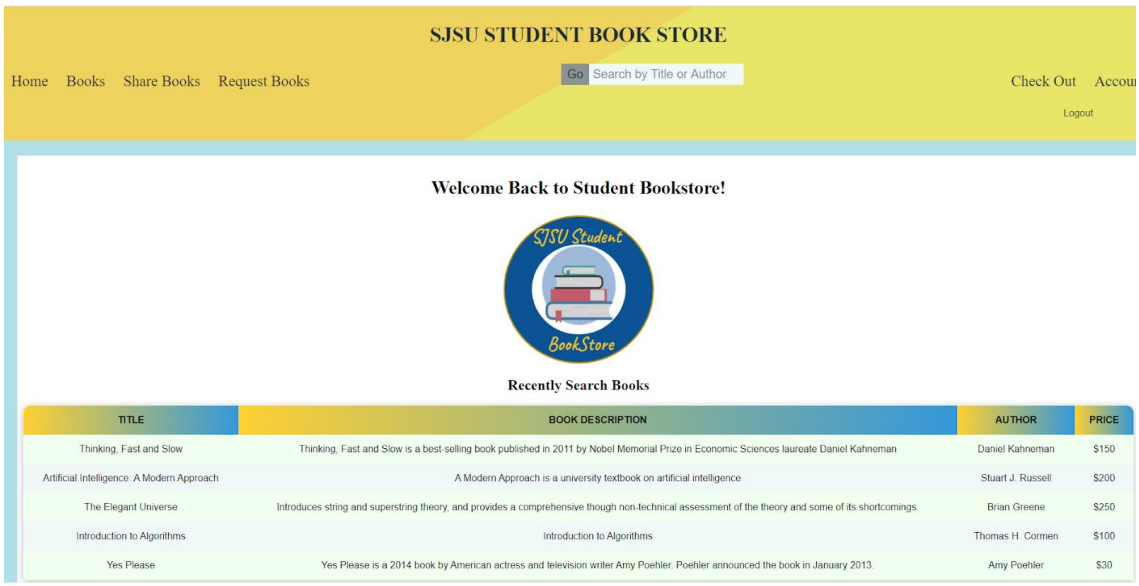
The above screenshot shows code that inserts items into the purchase table after user confirms purchase.

```
    <%
String db = "sjsu_textbookstore";
String user;
user = "root";
String password = "N00bcakes";

try {
 /**
 Establish connection with the database and selects all the transactions made by a user.
 Table shows title, transaction id, and price
 **/
    java.sql.Connection con;
    Class.forName("com.mysql.jdbc.Driver");
    con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false",user, password);
    String query = ("SELECT DISTINCT purchase.transaction_id, book.title, book.price"
            +  "FROM purchase,book WHERE account_id = ? AND purchase.book_id = book.book_id");
    String account_id=(String)request.getSession().getAttribute("userid");
    PreparedStatement prepared = con.prepareStatement(query);
    prepared.setString(1, account_id);
    ResultSet rs = prepared.executeQuery();
    rs = prepared.executeQuery();
    while (rs.next()) {
        out.println("<tr> <td>" + rs.getString("transaction_id") + "</td> <td>" +
        rs.getString("Title") + "</td><td>" + rs.getString("Price") + "</td></tr>");
    }
    rs.close();
    con.close();

} catch(SQLException e) {
    out.println("SQLException caught: " + e.getMessage());
}

%>
```

The screenshot above selects distinct purchases and displays the results in the table.

Recent Search

Webpage



The screenshot above shows the recent search history of a member who is logged in. The member must search for a book by author or name from the book page, the recent search shows the most 5 recent search results.

Table

```
mysql> describe recent_search;
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| search_id  | int  | NO   | PRI | NULL    | auto_increment |
| account_id | int  | NO   | PRI | NULL    |                |
| book_id    | int  | NO   | PRI | NULL    |                |
+------------+------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.recent_search;
+-----------+------------+---------+
| search_id | account_id | book_id |
+-----------+------------+---------+
|         1 |          1 |       1 |
|         2 |          1 |       9 |
|         3 |          1 |       4 |
|         4 |          1 |      11 |
|         5 |          1 |       5 |
|         6 |          1 |       7 |
|         7 |          2 |       8 |
|         8 |          2 |      14 |
|         9 |          3 |       9 |
|        10 |          3 |      10 |
|        11 |          3 |       2 |
|        12 |         14 |       2 |
|        13 |         14 |       7 |
|        14 |         11 |       8 |
|        15 |          5 |      14 |
+-----------+------------+---------+
15 rows in set (0.00 sec)
```

Frontend Code

```
<h2>Recently Search Books</h2>
<table class="table-desgin">
    <tr>
        <th>Title</th>
        <th>Book Description</th>
        <th>Author</th>
        <th>Price</th>
    </tr>
```

The above code will create a table with four columns and store the recently search books for members.

Backend Code

```
<%
 String db = "sjsu_textbookstore";
 String user;
 user = "root";
 String password = "N00bcakes";

 try {
java.sql.Connection con;
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false",user, password);
String query = ("SELECT * FROM recent_search,book WHERE account_id = ?"
        + "AND recent_search.book_id = book.book_id order by search_id desc limit 5");
String account_id=(String)request.getSession().getAttribute("userid");
PreparedStatement prepared = con.prepareStatement(query);
prepared.setString(1, account_id);
ResultSet rs = prepared.executeQuery();
rs = prepared.executeQuery();
while (rs.next()) {
        out.println("<tr> <td>" + rs.getString("title") + "</td> <td>" +
        rs.getString("description") + "</td><td>" + rs.getString("author") + "</td><td>$" + rs.getString("price") +"</td></tr>");
}
//Close connections
rs.close();
con.close();

} catch(SQLException e) {
   out.println("SQLException caught: " + e.getMessage());
}

%>
```

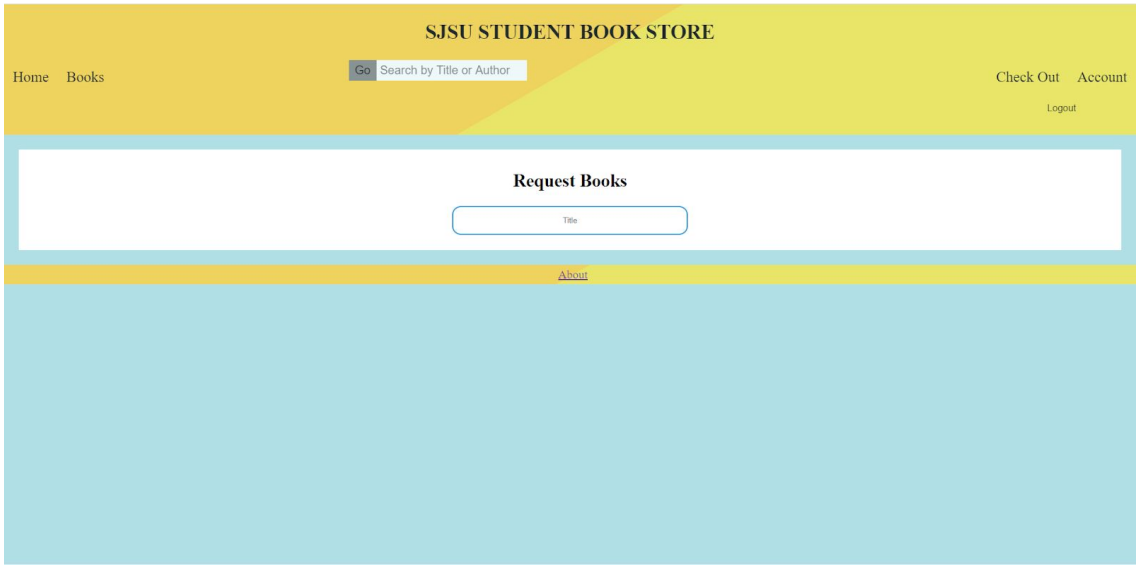
The above code will select from recent search table and display the five most recently searched for each member. Some html code is embedded in this statement.

```
            String account_id = (String) request.getSession().getAttribute("userid");
            String query = "INSERT INTO recent_search (book_id,account_id) VALUES ('" + rs.getInt("book_id") + "', '"
                    + account_id + "')";
            PreparedStatement prepared = con.prepareStatement(query);
            prepared.execute();
```

The above code will insert into recent search after a user has searched for a book on the search tab.

Request Books

Webpage



The screenshot above shows the request books page, where members can request for books that aren't present in the database, by inputting the title of the book. The requests are viewable by all the admins.

Table

```
mysql> describe request_books;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| request_id | int          | NO   | PRI | NULL    | auto_increment |
| account_id | int          | NO   | PRI | NULL    |                |
| title      | varchar(250) | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> Select * from sjsu_textbookstore.request_books;
+------------+------------+---------------------------------+
| request_id | account_id | title                           |
+------------+------------+---------------------------------+
|          1 |          1 | Tao Te Ching                    |
|          2 |          5 | Meditations                     |
|          3 |          5 | Nicomachean Ethics              |
|          4 |          2 | The Art of War                  |
|          5 |          2 | Trainspotting                   |
|          6 |          2 | Gravity's Rainbow               |
|          7 |         14 | The Odyssey                     |
|          8 |         14 | The Lord of the Rings Trilogy   |
|          9 |         14 | Dubliners                       |
|         10 |          3 | Jurassic Park                   |
|         11 |          3 | One Flew Over the Cuckoo?s Nest |
|         12 |          3 | The Road                        |
|         13 |          7 | Fahrenheit 451                  |
|         14 |          7 | A Confederacy of Dunces         |
|         15 |          7 | Hard Times                      |
+------------+------------+---------------------------------+
15 rows in set (0.00 sec)
```

Frontend Code

```
<section>
    <h1>Request Books</h1>
    <!-- Request books form for user -->
    <form  action = "./request_books.jsp" method = "POST" class="input-group-1" >
        <input class = "input-field-1" type= "text" placeholder="Title" name= "title" />
    </form>
```

The above code displays a form asking for the user to provide a title of a book that they want to upload.

Backend Code

```
<% //Enters the book that user requested into request books table
 String title = request.getParameter("title");
 String users;
 users = "root";
 String passwords = "N00bcakes";

 try {
        java.sql.Connection con;
        Class.forName("com.mysql.jdbc.Driver");
        con =
        DriverManager.getConnection("jdbc:mysql://localhost:3306/sjsu_textbookstore?autoReconnect=true&useSSL=false",users, passwords);

        if(title != null)
        {
            String account_id=(String)request.getSession().getAttribute("userid");
            String query = "INSERT into request_books (account_id, title) values (?,?)";
            PreparedStatement prepared = con.prepareStatement(query);
            prepared.setString(1, account_id);
            prepared.setString(2, title);
            prepared.executeUpdate();
            //Close connection
            con.close();
        }
 } catch(SQLException e) {
        out.println("SQLException caught: " + e.getMessage());
    }

%>
```

The above code inserts the request books into request_books tables after the user fills out the form that was screenshotted above.

Project Conclusion

Lessons Learned

Adrian Chan

Working on this project has helped me learn more about how front and back end development works. Embedding MySql in the backend with Java was new to me and was also one of the favorite parts of the project. I also learned more about the front end language of html and css. Prior to this project I did not know that html code could be embedded in the print statements of the backend code as well. Learning about this new feature allows me to improve the display of the webpage after data is retrieved from the database. Overall this project has taught me alot about the three-tier architecture used in web programming and also working as a team.

Bikalp Timalsina

This project has helped me learn a lot about full stack development. The career path I was interested in was Full Stack Development and while working on this project, it helped me understand lots of fundamental concepts which I never had the knowledge before. It was a good experience to improve my knowledge, on JSP, JAVA, SERVLET, JDBC and HTML5/CSS. All of these skills which I never knew and working on this project helped me develop a whole set of different types of skills needed for my career. Helped me understand more about teamwork and how to complete tasks as the part of the group. I learned to manage my time well and work on a pace where everyone was. I learned to communicate better with my group throughout the project, since communication is a key part of team work.

I learned to design webpages using HTML5/CSS, I have never used it before and now I feel comfortable applying my knowledge in future projects. I am also much more confident with my backend skills, since now I know a lot more about database management systems.

Jennifer Yang

From this project, I learned more about how web applications work and its architecture. I got to write codes that were able to pull the data from the database and present the data in the web application. This was my first time using a database for a web application and it was great. I had experience in creating web applications but only static web applications. It was very exciting to use a database and personalize a dynamic web application according to its user.

Future Improvements

For the future of the SJSU Student Bookstore, Team 6 envisions better features to improve user experience and performance. If there was time, the code for SJSU Student Bookstore would be cleaner. Below is a list of detailed list of requirements SJSU Student Bookstore would have like to incorporate:

Adding more javascript:

- **Adding javascript would make the webpage more interactive to users.**
- **This would provide a better GUI for users. For example, when you confirm a purchase you can use buttons instead of text fields.**
- **Adding images to the books**

Functionality to add people:

- **Members can search for other members and they can add them.**
- **With the addition of that they can also chat with other users whom they added as a friend.**

Adding Security:

- **Protecting against SQL injection**
- **2 step authentication**
- **Protecting against XSS attacks**
- **Providing error messages as a GIF**
- **Use HTTPS protocol to have security over the internet**

Improving Search Bar:

- **Allow members and guests to filter results with more constraints and not just by title and author.**
- **Members can limit the size of the search result.**