Book Nook

Maegan Lucas
April 27, 2023

1 Introduction

Book Nook is a reader review application. Book Nook is for both authors and readers. Both authors and readers can log-in to the application with a username and password. Based on the type of user, a role is assigned. These roles will determine which permissions are given to the user. Authors are able to enter data about their books including, but not limited to, book title, author's name, series, volume, International Standard Book Number (ISBN), publisher, genre, categories, and synopsis and then save the book to the database. They can also modify their book entries or delete them entirely. When the author chooses to modify the book, the modification screen will show the information prefilled that can then be changed. Once the book is entered into the database, readers can search the database for books to add them to their collection and the reader can choose whether the book is one they have read, one they are currently reading, or one they would like to read in the future.

For books a reader has marked as read, the reader can then enter their review. This includes the date the reader started the book, the date the reader finished the book, their numerical rating of the book for multiple categories, and a written review of the book. They can also mark any review category as null if they choose to. For books a reader has marked as currently reading, the book will be added to the reader's "Currently Reading" list. For books marked to be read in the future, the book will be added to the reader's "To Be Read" list. These book statuses can be changed by the reader at any time or removed entirely.

Readers will be able to pull reports about the books they've read over a period of time designated by the Readers. This report will contain a breakdown of the genres and categories of the books they have read. The report will also contain the number of books marked as read during the time period as well as the number of words read. This report will also contain a list of all the books read within this time frame. It will also contain a rating breakdown of each category of the books read within the time period.

Authors will be able to pull reports about their book's reviews. They can pull reports for individual books that show the author a breakdown of the ratings their book received from the readers. This breakdown of ratings will be broken down into each review category.

The design of Book Nook allows for users to easily pull information about books for their reviews. With this design, the review will use the ISBN from the book to be able to gain the other information needed such as the title and the author. This design will also help with giving the author a report on a certain book in the same manner by being able to pull all reviews with the same ISBN.

The database for Book Nook stores all of the information necessary to use Book Nook as an author or a reader. It allows for updating, creating, and deleting of information about books, reviews, and users. It also allows information from the database to be pulled and displayed for both readers and authors. For authors, it can pull all the books written by that author as well as all of the reviews for a certain book. For readers, it can pull the books marked by the reader based on its reading status. It can also pull all of the reviews from a reader specified time range.

A Java application allows the user to interact with the program and the database through a variety of buttons, text fields, drop down menus, and more. Authors and readers have different home screens as well as functionalities available to them. Authors cannot see reader pages and vice versa. The application will also show popups if there are errors in registering, logging in, or entering ISBN numbers. Readers can search books that are entered by the author into the database, set or edit their reading status, delete their reading status, or enter a review for the read books. Confirmation alerts are also used when deleting a profile or creating an account.

2 Deliverable 1: Project Description

When you read a book, you tend to decide how much you liked the book based on different areas like storyline, characters, setting, writing style, and much more. Readers like to take their review of a book and store that review somewhere that they can look back on later. This helps readers decide if they would read the book again, recommend the book to other readers, or read other books written by the same author. However, most book review platforms are very limited in how you can rate a book, with most only allowing a general overview of an entire book, not broken down into different categories. Book Nook allows users to rate their books on more than one category. Readers also like to see the breakdown of what they have read over time. This is not a feature of most book review platforms. Book Nook will allow readers to pull a report of the books they have read over a desired period of time broken down into many areas such as the list of books read, the breakdown of genres, and how many of each review score were given. Authors will also be able to see a breakdown of the reviews across each category for their books.

A database will be the best way to implement Book Nook. A database will be able to store the logins for the authors and the readers. When authors add their books, the books are stored in the database. The readers are then able to search the database of books to find the one they want to read, are reading, or have read. A database also allows authors to manipulate their book entries or delete them entirely.

When readers have finished a book, they are able to review their book. Once their review is completed, this review is stored into a database. This allows the reader to be able to pull their reviews for a report for a designated period of time and show a breakdown of their review scores across the categories. This also allows the author to pull a report on the reviews stored for their books and show a breakdown of the review scores across the categories.

2.1 User Classes

Readers: Someone who intends to use the system to be able to store their reviews of the books they have read, keep track of the book(s) they are currently reading, and create a list of books they would like to read.

Authors: Someone who intends to use the system to add their books to the database to allow for reviews. They can add, modify, or delete their books at any time.

2.2 Stored Information

User Information: User’s username and password are stored along with their respective role – author or reader. These roles determine their permissions within the application.

Books: Books are added to the database by authors with their information such as their title, author’s name, series, volume, ISBN, genre, category, and synopsis. Readers are then able to assign them a status of read, reading, or to be read.

Reviews: Once readers have read a book, they are able to give a numerical review based on different categories such as storyline, characters, setting, writing style, and more as well as a written review of their thoughts. These reviews are able to be pulled by both readers and authors for reports in different capacities.

2.3 Database Operations Needed

Authors shall enter information regarding a book they wrote including its title, author, series, volume, ISBN, publisher, genre, categories, synopsis, and number of pages. Upon entering the book, it will appear in a list of stored books.

Readers shall be able to query the books to find a specific book to add to their collection. The result will be a list of books.

Readers shall be able to select a reading status of read for a book and enter information for a review including date started, date finished, a numerical rating on different categories, and a written review. Upon entering the review, it will be stored in a list of reviews.

Readers shall be able to select a reading status of reading for a book and enter information including current page number. Upon entering this status, the book will appear in a list of books with the same status.

Readers shall be able to select a reading status of to be read. Upon selecting this status, the book will appear in a list of books with the same status.

Authors shall be able to modify books after they are saved into the database. Upon modifying the book's attribute information, the book will show with the new information in the list of stored books.

Readers shall be able to modify reviews after they are saved. Upon modifying the review information, the review shall show with the new information in the list of reviews.

Readers shall be able to delete books of any status in their collection. Upon deletion, the books will be removed from their respective status list and the collection of books.

Authors shall be able to delete books from the database at any time. Upon deletion, the books will be removed from the stored list of books.

2.4 Mockup of Application

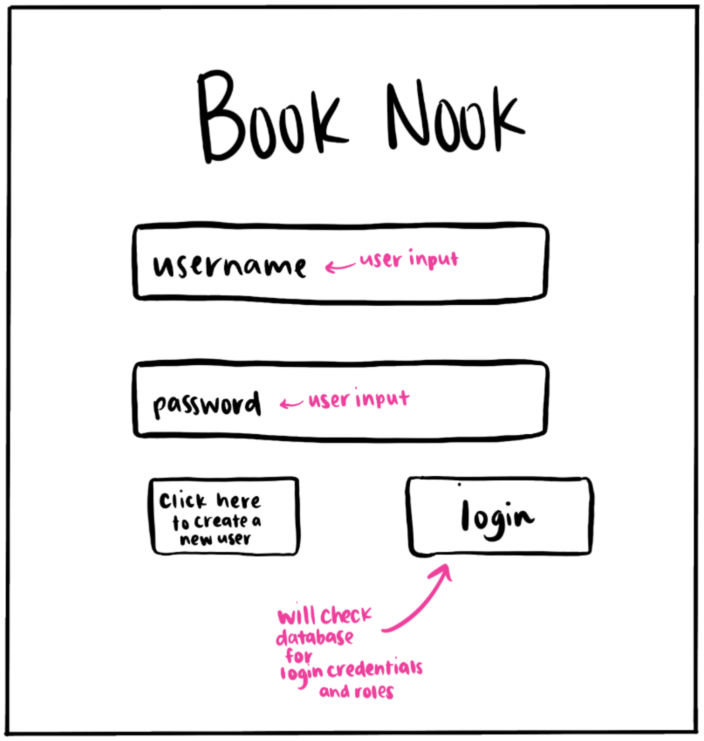

Book Nook
username
user input
password
user input
Click here to create a new user
login
will check database for login credentials and roles

Figure 1: A mockup of the home screen of Book Nook seen by all users.

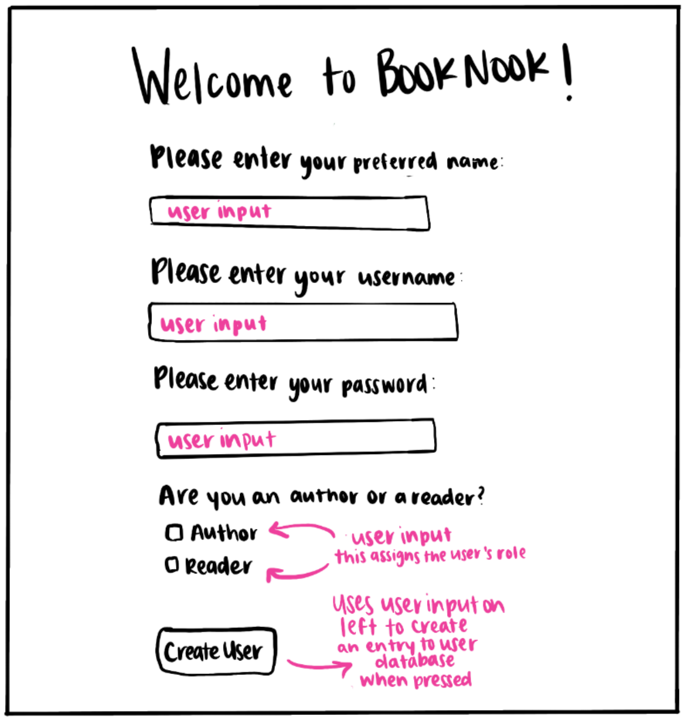

Welcome to Book Nook!
Please enter your preferred name:
user input
Please enter your username:
user input
Please enter your password:
user input
Are you an author or a reader?
Author
Reader
user input
this assigns the user's role
Create User
Uses user input on left to create an entry to user database when pressed

Figure 2: A mockup of the user registration screen of Book Nook seen by all users.

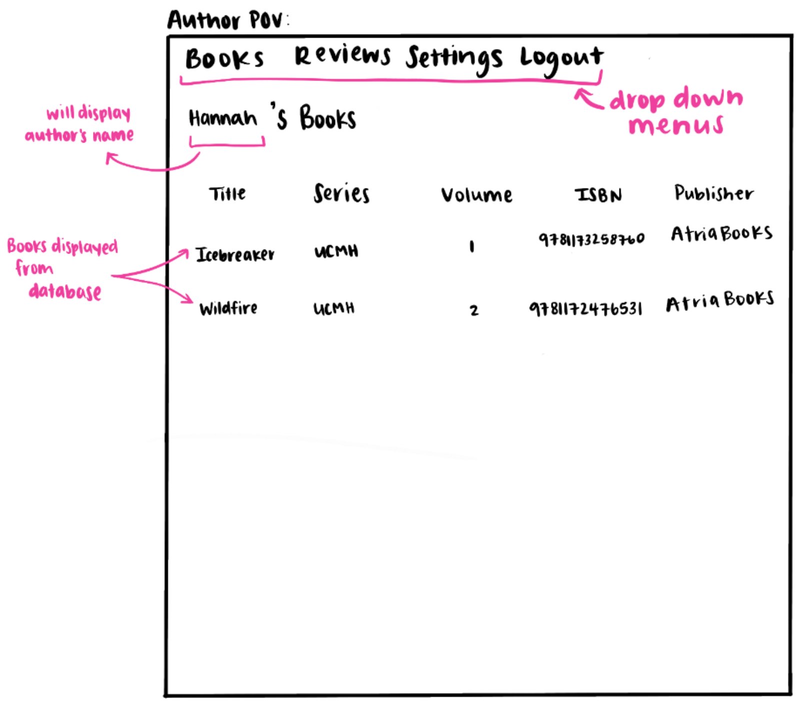


Figure 3: A mockup of the home screen of an Author user of Book Nook.

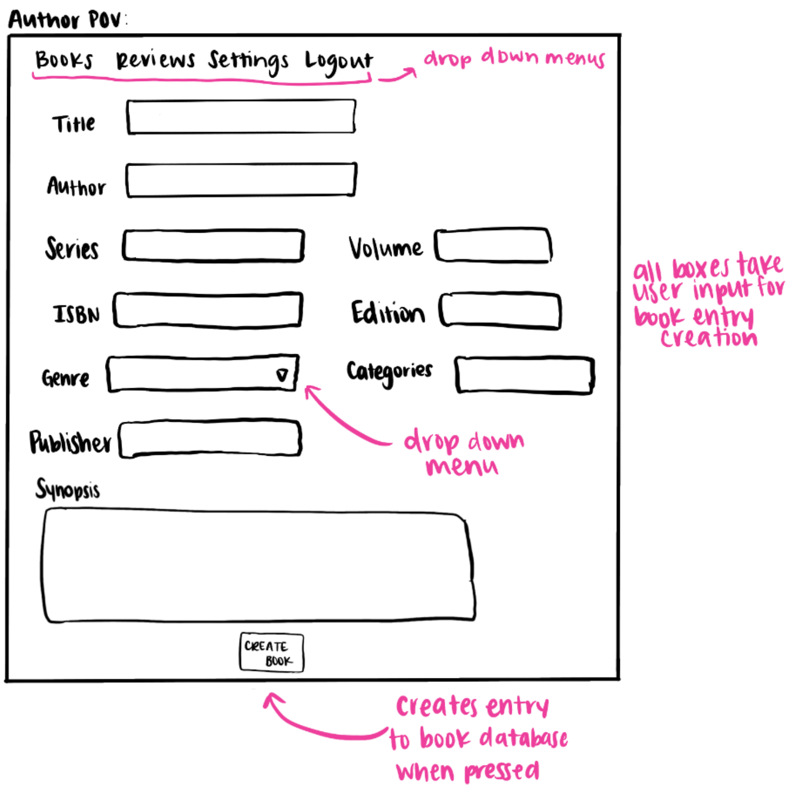

Author POV:
Books Reviews Settings Logout
drop down menus
Title
Author
Series
Volume
ISBN
Edition
Genre
Categories
Publisher
Synopsis
all boxes take user input for book entry creation
drop down menu
CREATE BOOK
creates entry to book database when pressed

Figure 4: A mockup of the book entry creation screen seen by Author users of Book Nook.

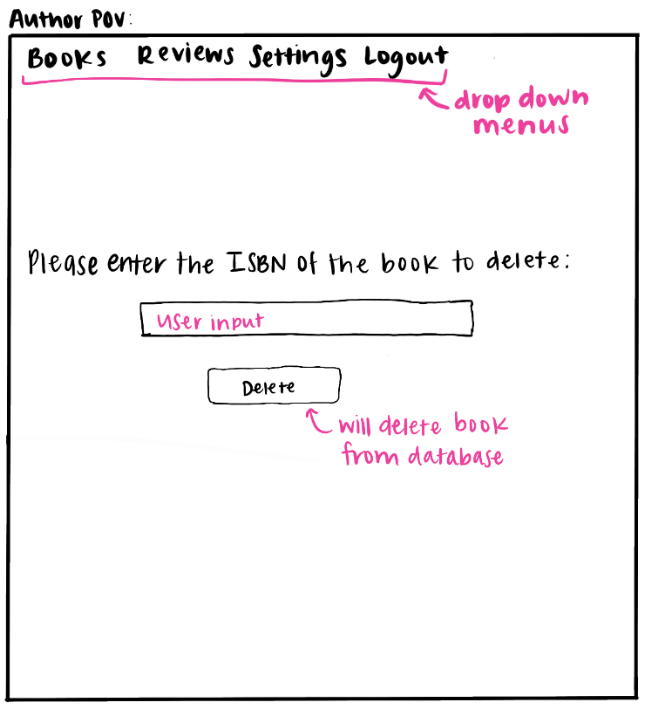
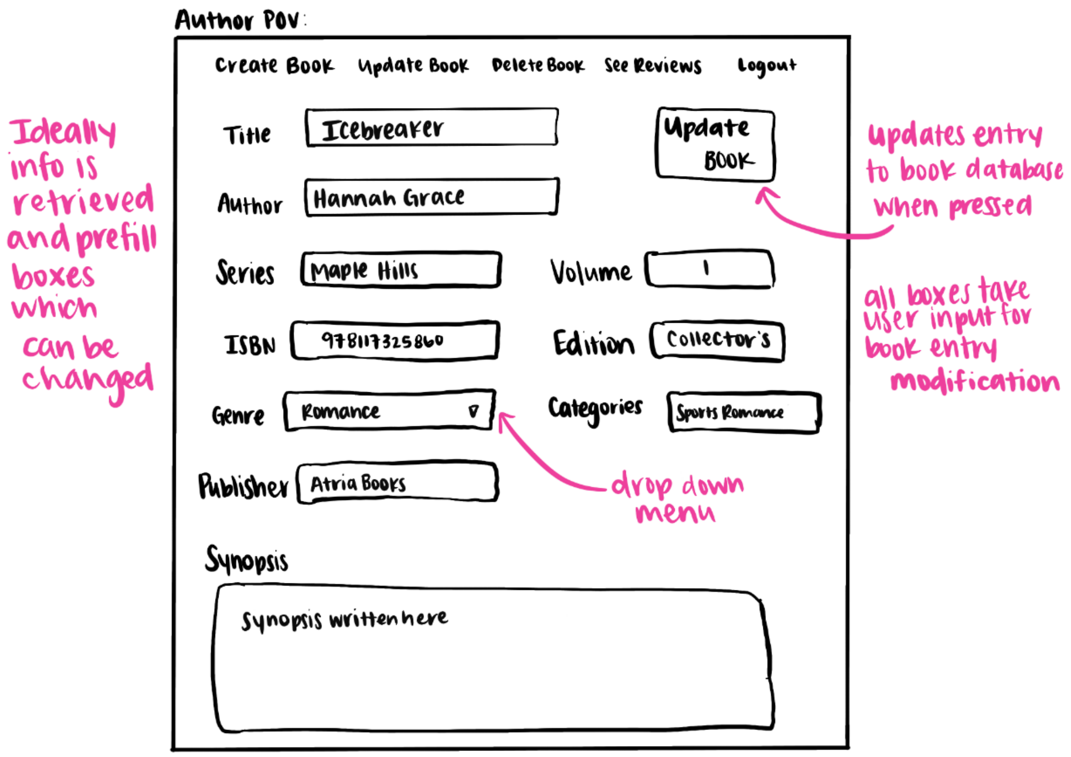

Author POV:
Books
Reviews
Settings
Logout
drop down menus
Please enter the ISBN of the book to delete:
user input
Delete
will delete book from database

Figure 5: A mockup of the book deletion page seen by Author users of Book Nook.


Author POV:
Create Book
Update Book
Delete Book
See Reviews
Logout
Update Book
updates entry to book database when pressed
Ideally info is retrieved and prefill boxes which can be changed
all boxes take user input for book entry modification
drop down menu
Synopsis

Title: Icebreaker
Author: Hannah Grace
Series: Maple Hills
Volume: 1
ISBN: 9781117325860
Edition: Collector's
Genre: Romance
Categories: Sports Romance
Publisher: Atria Books
Synopsis written here

Figure 6: A mockup of the book modification screen seen by an Author user of Book Nook.

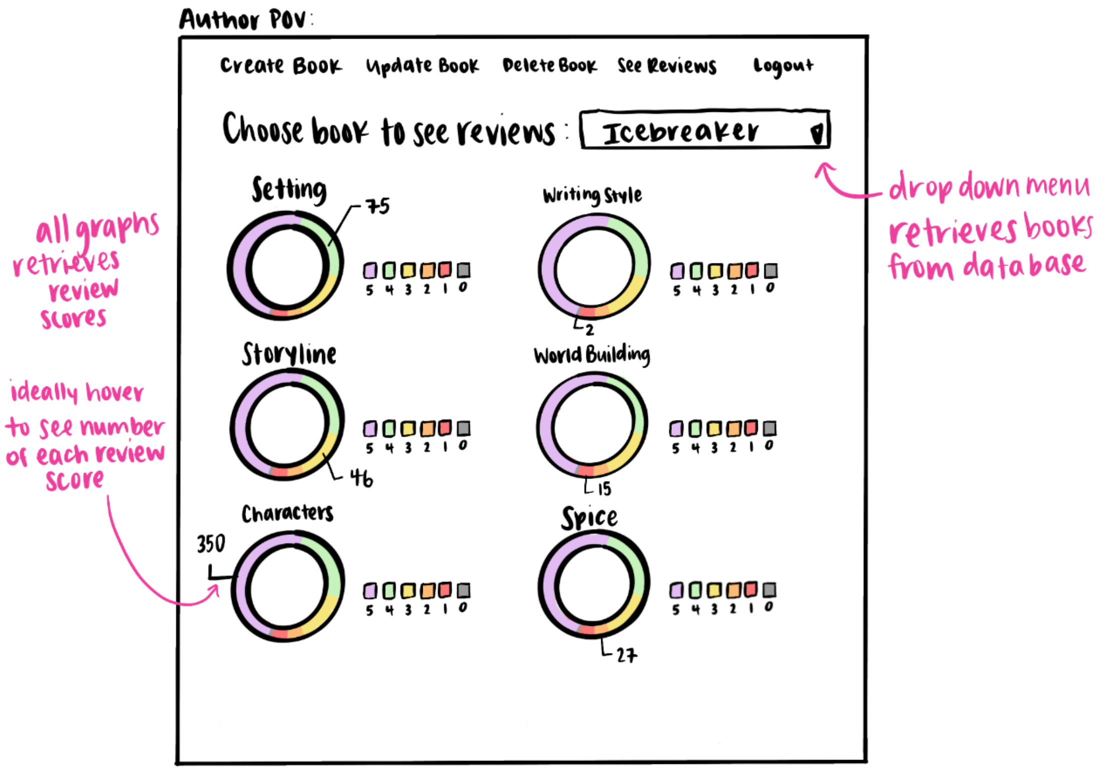
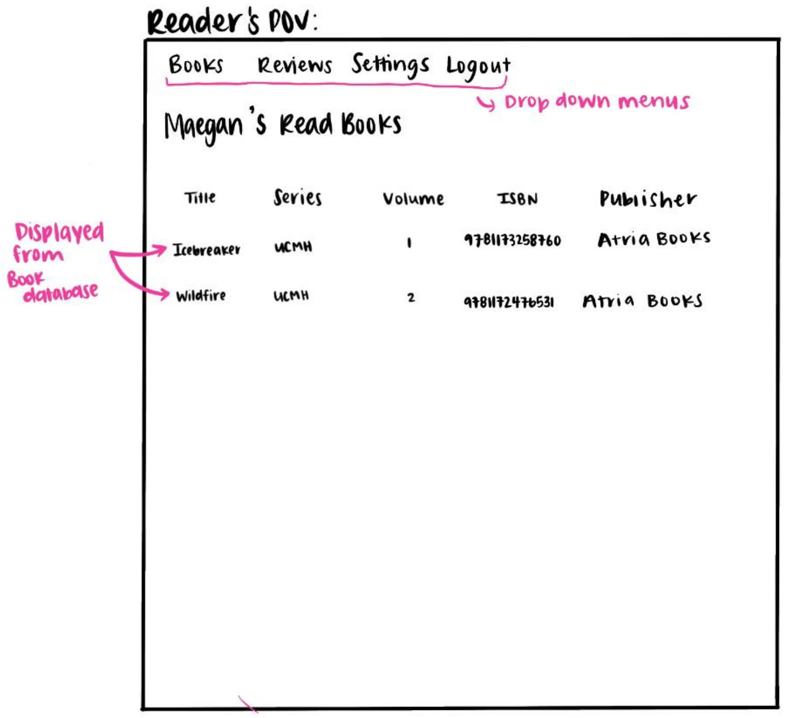
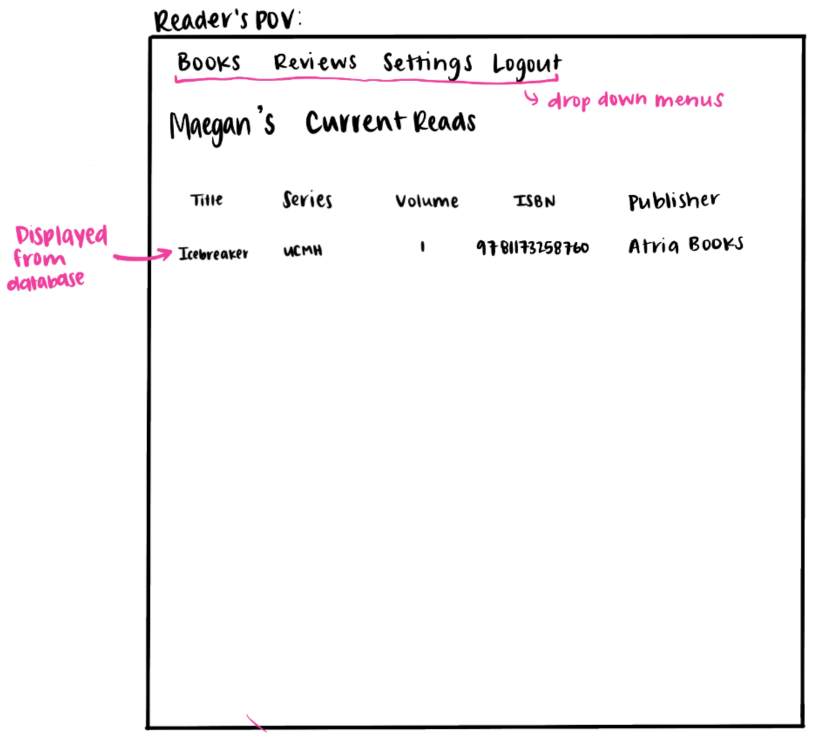

Author POV:
Create Book
Update Book
Delete Book
See Reviews
Logout
Choose book to see reviews:
Icebreaker
drop down menu retrieves books from database
all graphs retrieves review scores
ideally hover to see number of each review score
Setting
75
Writing Style
2
Storyline
46
World Building
15
Characters
350
Spice
27
5 4 3 2 1 0

Figure 7: A mockup of the report screen seen by Author users of Book Nook.



Title	Series	Volume	ISBN	Publisher
Icebreaker	UCMH	1	9781173258760	Atria Books
Wildfire	UCMH	2	9781172476531	Atria Books

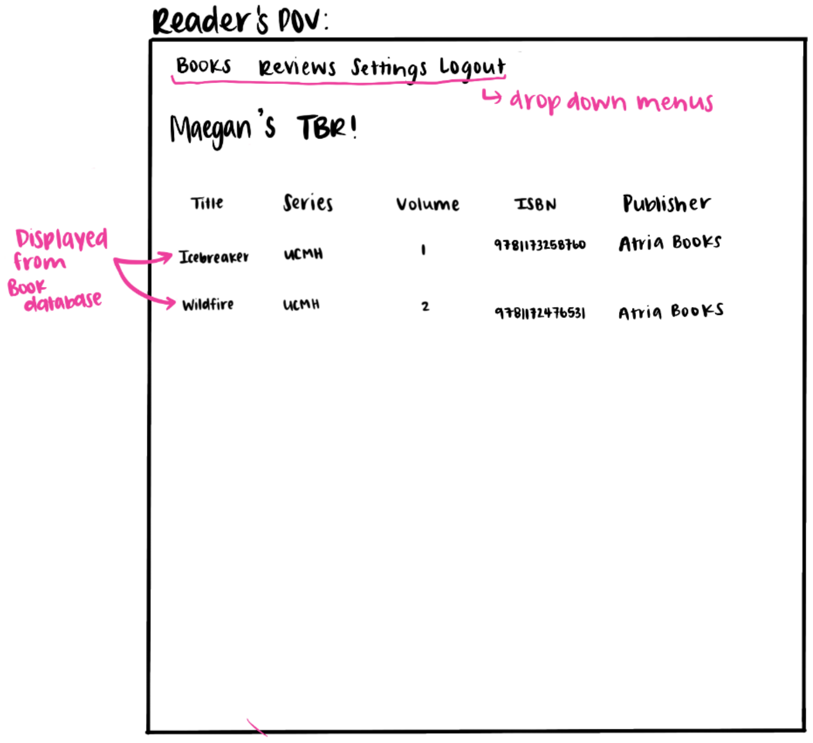
Figure 8: A mockup of the home screen seen by a Reader user of Book Nook. Contains a list of the books they have marked as read.



Maegan's Current Reads

Title	Series	Volume	ISBN	Publisher
Icebreaker	UCMH	1	9781173258760	Atria Books

Figure 9: A mockup of the Currently Reading screen seen by a Reader user of Book Nook.



Maegan's TBR!

Title	Series	Volume	ISBN	Publisher
Icebreaker	UCMH	1	9781173258760	Atria Books
Wildfire	UCMH	2	9781172476531	Atria Books

Figure 10: A mockup of the To Be Read screen seen by a Reader user of Book Nook.

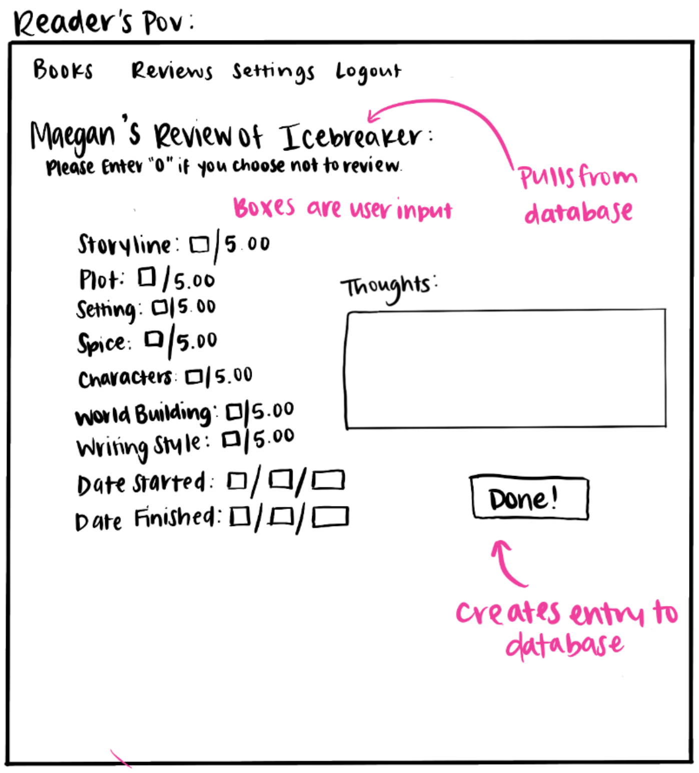


Figure 11: A mockup of the review screen seen by a Reader user of Book Nook.

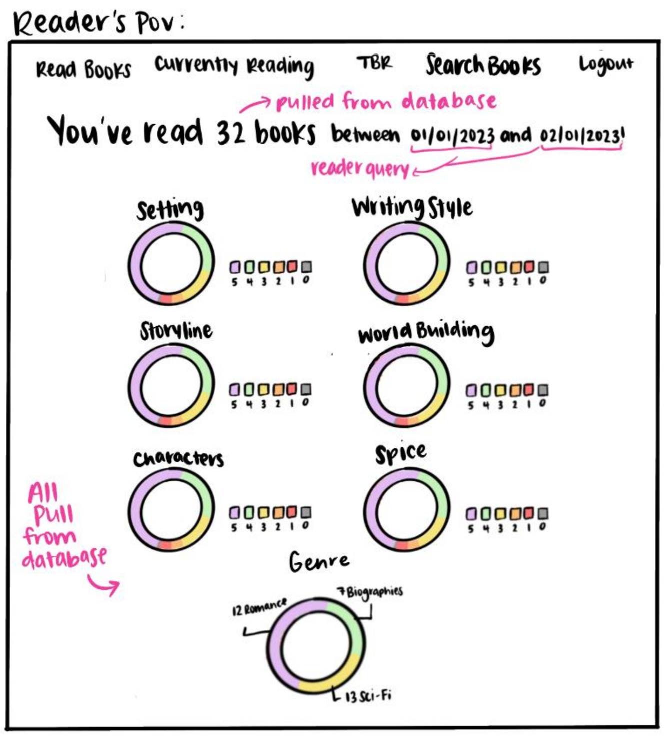


Figure 12: A mockup of the report screen seen by a Reader user of Book Nook.

3 *Deliverable 2:* Design of Relations

In this section, the design of the relational database is described. First an E/R diagram models our problem. We translate the model into a relational schema. Analysis demonstrates that the schema is in 3rd normal form (3NF).

3.1 E/R Modeling

The Entity/Relationship Model (E/R Model) for Book Nook can be found in Figure 13 below. It contains the entity Reader, which is connected in a many-to-many relationship with the entity Books called Reads. The Readers entity is also connected with the Reviews entity in a many-to-many relationship called Writes Review. The entity Authors is also connected to the Books entity in a many-to-many relationship called Writes Book. The entity Authors is connected to the Reviews entity with a many-to-many relationship called Views Review. The Reviews entity is connected to the Books entity with a many-to-many relationship called Review Of. In this E/R Model, the attributes wsRating and wbRating can be found in the Reviews entity. These stand for "Writing Style Rating" and "World Building Rating" respectively.

For Book Nook's design, it was decided that the Reviews entity should be made a weak entity relation. This allows it to pull the key attribute, the ISBN, from the book being reviewed. From this ISBN, the title and author of the book can be joined with the review's data. Another design decision for Book Nook was to have the Reads relationship have its own attribute of *readingStatus*.

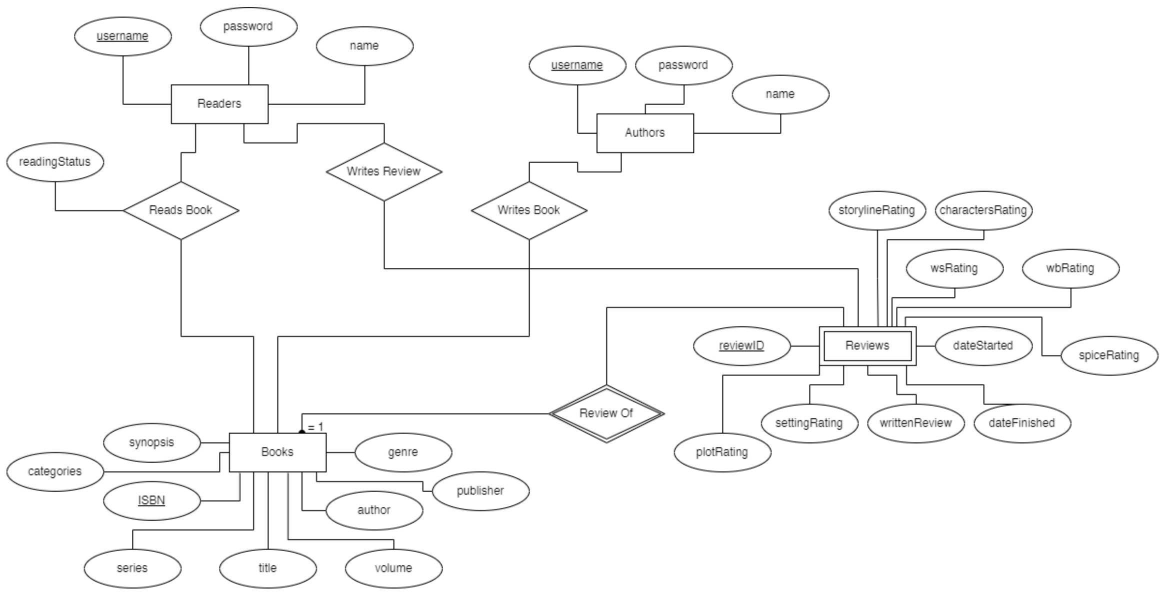


Figure 13: The Entity Relation Model for Book Nook.

3.2 Relational Schema

The database schema for Book Nook consists of the following relation schemas and their respective data types:

- Readers(<u>username</u>, password, name)
  - username: CHAR(30)
  - password: CHAR(30)
  - name: CHAR (25)
- Authors(<u>username</u>, password, name)
  - username: VARCHAR(30)
  - password: VARCHAR(30)
  - name: VARCHAR(25)
- Books(<u>ISBN</u>, title, author, series, volume, publisher, genre, categories, synopsis, authorUsername)
  - ISBN: CHAR(13)
  - title: VARCHAR(250)
  - author: VARCHAR(100)
  - series: VARCHAR(100)
  - volume: DECIMAL(2, 1)
  - publisher: VARCHAR(50)
  - genre: VARCHAR(50)
  - categories: VARCHAR(100)
  - synopsis: CHAR(10000)
  - authorUsername: VARCHAR(30)
- Reviews(<u>reviewID</u>, ISBN, storylineRating, plotRating, settingRating, spiceRating, charactersRating, wbRating, wsRating, dateStarted, dateFinished, writtenReview)
  - reviewID: INTEGER
  - ISBN: CHAR(13)
  - storylineRating: DECIMAL(3, 2)
  - plotRating: DECIMAL(3, 2)
  - settingRating: DECIMAL(3, 2)
  - spiceRating: DECIMAL(3, 2)
  - charactersRating: DECIMAL(3, 2)
  - wbRating: DECIMAL(3, 2)
  - wsRating: DECIMAL(3, 2)
  - dateStarted: DATE
  - dateFinished: DATE
  - writtenReview: VARCHAR(1000)
- ReadsBook(<u>readerUsername</u>, <u>ISBN</u>, readingStatus)
  - readerUsername: VARCHAR(30)
  - ISBN: CHAR(13)
  - readingStatus: VARCHAR(7)

- WritesReview(<u>readerUsername</u>, <u>reviewID</u>)
    - readerUsername: VARCHAR(30)
    - reviewID: Integer

An analysis of Book Nook's schema determined the functional dependencies. For the Reader relation the functional dependency $Reader\ Username \rightarrow Password, Name$ was found. Since the reader's username is the primary key of this relation, this relation is in 3NF. Similarly, for the Author relation the functional dependency $Author\ Username \rightarrow Password, Name$ was found. Since the author's username is the primary key of this relation, this relation is in 3NF.

The Reviews relation contains the functional dependency $Review\ ID \rightarrow Book\ ISBN, Plot\ Rating, Storyline\ Rating, Character\ Rating, Setting\ Rating, Writing\ Style\ Rating, World\ Building\ Rating, Date\ Started, Date\ Finished, Written\ Review$. This relation is 3NF compliant as its left-hand side, $Review\ ID$, is a key of the relation.

The Books relation was found to contain the functional dependencies $Book\ ISBN \rightarrow Title, Author, Series, Volume, Publisher, Genre, Categories, Synopsis, Author\ Username$ and $Title, Author \rightarrow Series, Volume, Genre, Categories, Synopsis$. Since the primary key of the relation is the book's ISBN, this relation violates 3NF. The 3NF Synthesis Algorithm was performed on the relation to decompose it into a set of 3NF compliant relations. The results of this composition are:

- $BookAuthor(Book\ ISBN, Title, Author, Publisher, Author\ Username)$
- $BookDetails(Title, Author, Series, Volume, Genre, Categories, Synopsis)$

4 Deliverable 3: Implementation of Database

This section discusses the translation of the design into an actual database in SQL. It focuses upon the database exclusively and does not include detailed design/implementation of the application that utilizes the database.

4.1 Definition of SQL Database Schema

This section presents the SQL schema for the database designed in the previous section.

4.1.1 Database schema

```
create table if not exists Readers(

username VARCHAR(30) primary key,

password VARCHAR(30),

name VARCHAR (25)

);

create table if not exists Authors(

username VARCHAR(30) primary key,

password VARCHAR(30),

name VARCHAR(25)

);

create table if not exists BookAuthor(

ISBN CHAR(13) primary key,

title VARCHAR(250),

author VARCHAR(100),

publisher VARCHAR(50) default null,

authorUsername VARCHAR(30) default null

);

create table if not exists BookDetails(
title VARCHAR(250),

author VARCHAR(100),

series VARCHAR(100) default null,

volume DECIMAL(2,1) default null,

genre VARCHAR(50) default null,

categories VARCHAR(100) default null,

synopsis VARCHAR(10000) default null

);

create table if not exists Reviews(

reviewID INTEGER,

ISBN CHAR(13),

storylineRating DECIMAL(3,2) default 0.00,

plotRating DECIMAL(3,2) default 0.00,

settingRating DECIMAL(3,2) default 0.00,

spiceRating DECIMAL(3,2) default 0.00,

charactersRating DECIMAL(3,2) default 0.00,

wbRating DECIMAL(3,2) default 0.00,

wsRating DECIMAL(3,2) default 0.00,

dateStarted DATE default null,

dateFinished DATE default null,

writtenReview VARCHAR(1000) default null,

primary key(reviewID, ISBN)

);

create table if not exists ReadsBook(

username VARCHAR(30),
ISBN CHAR(13),

readingStatus VARCHAR(7),

primary key(username, ISBN)

);

create table if not exists writesReview(

username VARCHAR(30),

reviewID INTEGER,

primary key(username, reviewID)

);
```

4.1.2 Sample Data

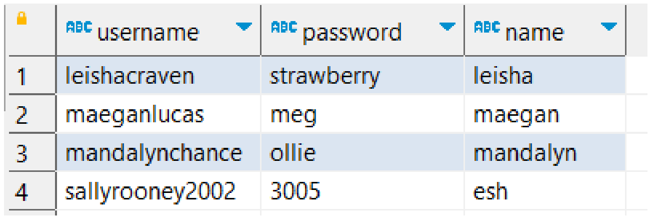
```
insert into Readers values

('maeganlucas', 'meg', 'maegan'),

('sallyrooney2002', '3005', 'esh'),

('mandalynchance', 'ollie', 'mandalyn'),

('leishacraven', 'strawberry', 'leisha');

insert into Authors values

('hannahgrace', '123', 'hannah'),

('rickriordan', '456', 'rick'),

('anahuang', '789', 'ana'),

('kieracass', '246', 'kiera'),

('taylorjreid', '345', 'taylor');

insert into BookAuthor values

('9781915593009', 'Icebreaker', 'Hannah Grace', 'Pig & Bear', 'hannahgrace'),

('9780545614948', 'The Selection', 'Kiera Cass', 'Scholastic Press',
'kieracass'),
('9780786838653', 'The Lightning Thief', 'Rick Riordan', 'Hyperion Books for
Children', 'rickriordan'),

('9798737416119', 'Twisted Love', 'Ana Huang', 'Boba Press', 'anahuang'),

('9781501161933', 'The Seven Husbands of Evelyn Hugo', 'Taylor Jenkins Reid',
'Washington Square Press', 'taylorjreid');

insert into BookDetails values

('Icebreaker', 'Hannah Grace', 'UCMH', 1.0, 'romance', 'sports, hockey',
'Anastasia Allen has worked her entire life for a shot at Team USA.

A competitive figure skater since she was five years old, a full college
scholarship thanks to her place on the Maple Hills skating team, and a
schedule that would make even the most driven person weep, Stassie comes to
win.

No exceptions.

Nathan Hawkins has never had a problem he couldn’t solve. As captain of the
Maple Hills Titans, he knows the responsibility of keeping the hockey team on
the ice rests on his shoulders.

When a misunderstanding results in the two teams sharing a rink, and
Anastasia’s partner gets hurt in the aftermath, Nate finds himself swapping
his stick for tights, and one scary coach for an even scarier one.

The pair find themselves stuck together in more ways than one, but it’s fine,
because Anastasia doesn’t even like hockey players…right?'),

('The Selection', 'Kiera Cass', 'The Selection', 1.0, 'romance', 'ya,
dystopian', 'For thirty-five girls, the Selection is the chance of a lifetime.
The opportunity to escape the life laid out for them since birth. To be swept
up in a world of glittering gowns and priceless jewels. To live in a palace
and compete for the heart of gorgeous Prince Maxon.

But for America Singer, being Selected is a nightmare. It means turning her
back on her secret love with Aspen, who is a caste below her. Leaving her home
to enter a fierce competition for a crown she doesn’t want. Living in a palace
that is constantly threatened by violent rebel attacks.

Then America meets Prince Maxon. Gradually, she starts to question all the
plans she’s made for herself—and realizes that the life she’s always dreamed
of may not compare to a future she never imagined.'),

('The Lightning Thief', 'Rick Riordan', 'Percy Jackson and the Olympians',
1.0, 'fantasy', 'ya, greek mythology', 'Percy Jackson is a good kid, but he
can’t seem to focus on his schoolwork or control his temper. And lately, being
away at boarding school is only getting worse - Percy could have sworn his
pre-algebra teacher turned into a monster and tried to kill him. When Percy’s
mom finds out, she knows it’s time that he knew the truth about where he came
```

```
from, and that he go to the one place he’ll be safe. She sends Percy to Camp
Half Blood, a summer camp for demigods (on Long Island), where he learns that
the father he never knew is Poseidon, God of the Sea. Soon a mystery unfolds
and together with his friends—one a satyr and the other the demigod daughter
of Athena - Percy sets out on a quest across the United States to reach the
gates of the Underworld (located in a recording studio in Hollywood) and
prevent a catastrophic war between the gods.'),

('Twisted Love', 'Ana Huang', 'Twisted', 1.0, 'romance', 'brother’s best
friend, millionaire', 'He has a heart of ice...but for her, he’d burn the
world.

Alex Volkov is a devil blessed with the face of an angel and cursed with a
past he can’t escape.

Driven by a tragedy that has haunted him for most of his life, his ruthless
pursuits for success and vengeance leave little room for matters of the heart.

But when he’s forced to look after his best friend’s sister, he starts to feel
something in his chest:

A crack.

A melt.

A fire that could end his world as he knew it.

***

Ava Chen is a free spirit trapped by nightmares of a childhood she can’t
remember.

But despite her broken past, she’s never stopped seeing the beauty in the
world…including the heart beneath the icy exterior of a man she shouldn’t
want.

Her brother’s best friend.

Her neighbor.

Her savior and her downfall.

Theirs is a love that was never supposed to happen—but when it does, it
unleashes secrets that could destroy them both…and everything they hold
dear.'),

('The Seven Husbands of Evelyn Hugo', 'Taylor Jenkins Reid', null, null,
'romance', 'lgbtq+, old hollywood', 'Aging and reclusive Hollywood movie icon
Evelyn Hugo is finally ready to tell the truth about her glamorous and
scandalous life. But when she chooses unknown magazine reporter Monique Grant
for the job, no one is more astounded than Monique herself. Why her? Why now?
```

```
Monique is not exactly on top of the world. Her husband has left her, and her
professional life is going nowhere. Regardless of why Evelyn has selected her
to write her biography, Monique is determined to use this opportunity to
jumpstart her career.

Summoned to Evelyn’s luxurious apartment, Monique listens in fascination as
the actress tells her story. From making her way to Los Angeles in the 1950s
to her decision to leave show business in the ‘80s, and, of course, the seven
husbands along the way, Evelyn unspools a tale of ruthless ambition,
unexpected friendship, and a great forbidden love. Monique begins to feel a
very real connection to the legendary star, but as Evelyn’s story near its
conclusion, it becomes clear that her life intersects with Monique’s own in
tragic and irreversible ways.');

insert into Reviews values

(1, '9781915593009', 5.00, 5.00, 4.75, 5.00, 5.00, 0.00, 5.00, '2022-09-25',
'2022-09-26', null ),

(2, '9780545614948', 5.00, 5.00, 5.00, 0.00, 4.00, 2.75, 3.00, '2023-01-01',
'2023-01-05', 'liked it'),

(3, '9781501161933', 5.00, 5.00, 5.00, 0.00, 5.00, 0.00, 5.00, '2023-02-22',
'2023-02-22', 'fantastic'),

(4, '9798737416119', 5.00, 5.00, 5.00, 5.00, 5.00, 0.00, 5.00, '2022-05-26',
'2022-06-12', null);

insert into ReadsBook values

('maeganlucas', '9781915593009', 'read'),

('sallyrooney2002', '9781501161933', 'read'),

('mandalynchance', '9798737416119', 'reading'),

('leishacraven', '9780786838653', 'tbr'),

('mandalynchance', '9781915593009', 'tbr'),

('maeganlucas', '9780545614948', 'read'),

insert into WritesReview values

('maeganlucas', 1),

('maeganlucas', 2),

('sallyrooney2002', 3),

('leishacraven', 4);
```

4.1.3 Schema Test Results

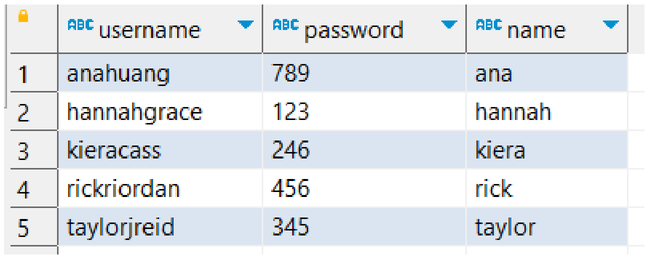
Result from SELECT * FROM Readers:



	username	password	name
1	leishacraven	strawberry	leisha
2	maeganlucas	meg	maegan
3	mandalynchance	ollie	mandalyn
4	sallyrooney2002	3005	esh

*Figure 14: The results of the SELECT * FROM Readers query.*

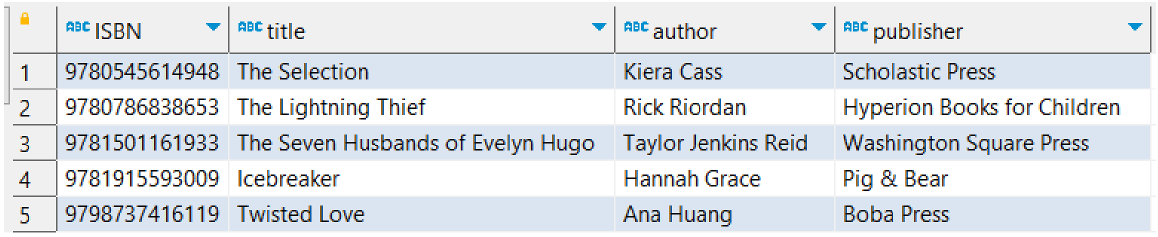
Result from SELECT * FROM Authors:



	username	password	name
1	anahuang	789	ana
2	hannahgrace	123	hannah
3	kieracass	246	kiera
4	rickriordan	456	rick
5	taylorjreid	345	taylor

*Figure 15: The results of the SELECT * FROM Authors query.*

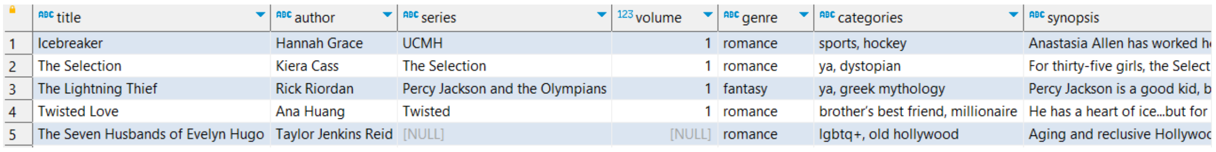
Results from SELECT * FROM BookAuthor:



	ISBN	title	author	publisher
1	9780545614948	The Selection	Kiera Cass	Scholastic Press
2	9780786838653	The Lightning Thief	Rick Riordan	Hyperion Books for Children
3	9781501161933	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	Washington Square Press
4	9781915593009	Icebreaker	Hannah Grace	Pig & Bear
5	9798737416119	Twisted Love	Ana Huang	Boba Press

*Figure 16: The results of the SELECT * FROM BookAuthor query.*

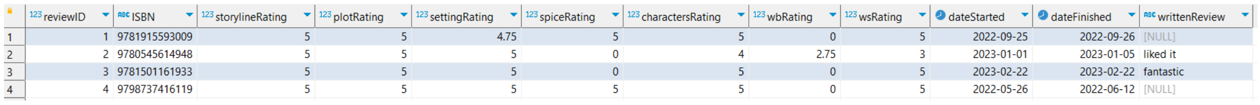
Results from SELECT * FROM BookDetails:



	title	author	series	volume	genre	categories	synopsis
1	Icebreaker	Hannah Grace	UCMH	1	romance	sports, hockey	Anastasia Allen has worked h
2	The Selection	Kiera Cass	The Selection	1	romance	ya, dystopian	For thirty-five girls, the Select
3	The Lightning Thief	Rick Riordan	Percy Jackson and the Olympians	1	fantasy	ya, greek mythology	Percy Jackson is a good kid, b
4	Twisted Love	Ana Huang	Twisted	1	romance	brother's best friend, millionaire	He has a heart of ice...but for
5	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	[NULL]	[NULL]	romance	lgbtq+, old hollywood	Aging and reclusive Hollywoo

*Figure 17: The results of the SELECT * FROM BookDetails query.*

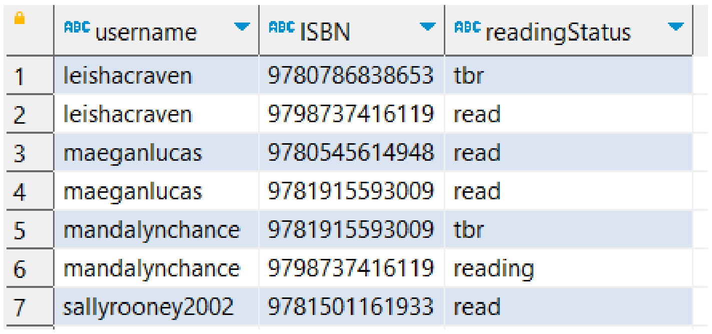
Results from SELECT * FROM Reviews:



	reviewID	ISBN	storylineRating	plotRating	settingRating	spiceRating	charactersRating	wbRating	wsRating	dateStarted	dateFinished	writtenReview
1	1	9781915593009	5	5	4.75	5	5	0	5	2022-09-25	2022-09-26	[NULL]
2	2	9780545614948	5	5	5	0	4	2.75	3	2023-01-01	2023-01-05	liked it
3	3	9781501161933	5	5	5	0	5	0	5	2023-02-22	2023-02-22	fantastic
4	4	9798737416119	5	5	5	5	5	0	5	2022-05-26	2022-06-12	[NULL]

*Figure 18: The results of the SELECT * FROM Reviews query.*

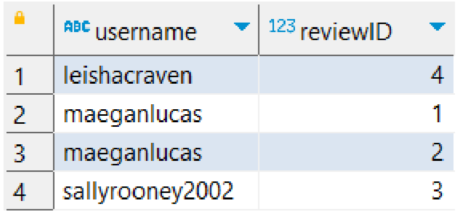
Results from SELECT * FROM ReadsBook:



	username	ISBN	readingStatus
1	leishacraven	9780786838653	tbr
2	leishacraven	9798737416119	read
3	maeganlucas	9780545614948	read
4	maeganlucas	9781915593009	read
5	mandalynchance	9781915593009	tbr
6	mandalynchance	9798737416119	reading
7	sallyrooney2002	9781501161933	read

*Figure 19: The results of the SELECT * FROM ReadsBook query.*

Results from SELECT * FROM WritesReview:



	username	reviewID
1	leishacraven	4
2	maeganlucas	1
3	maeganlucas	2
4	sallyrooney2002	3

*Figure 20: The results of the SELECT * FROM WritesReview query.*

4.2 Database Operations

Many operations will be performed on the Book Nook Database. These operations include:

- *Inserting Readers:* To insert a Reader user into the Book Nook database, an *INSERT* function will be used on the Readers table.
- *Inserting Authors:* To insert an Author user into the Book Nook database, an *INSERT* function will be used on the Authors table.
- *Inserting Books:* To insert a book into the Book Nook database, an *INSERT* function will be used on the BookAuthor and BookDetails tables.
- *Inserting Reviews:* To insert a review into the Book Nook database, an *INSERT* function will be used on the Review table.
- *Connecting Reviews to Readers:* To connect reviews to the reader that wrote them, an *INSERT* function will be used on the WritesReview table to enter the ReviewID and the username of the readers who wrote them.
- *Connecting Author to Books:* To connect books to their authors, an *INSERT* function will be used on the WritesBook table to enter the ISBN of the book and the author that wrote them.
- *Choosing Reading Status:* When a Reader chooses their reading status an *INSERT* function will be used to insert the reader's username, the book's ISBN, and the reading status into the table ReadsBook.
- *Updating Book Information:* To update a book's information, a *THETA JOIN* function will be used to join together the BookAuthor and BookDetails tables

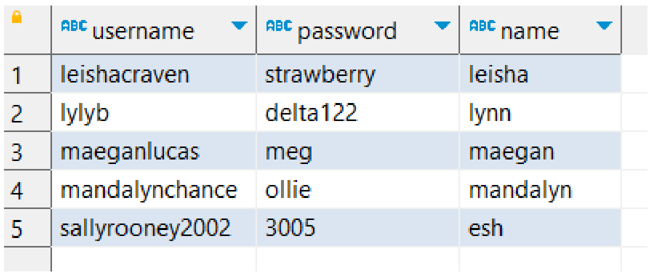
with the condition that BookAuthor’s title and author equals BookDetail’s title and author. An *UPDATE* function will then be used to the update the book’s information across both tables as necessary with the condition that the book’s ISBN matches.

- *Updating Readers:* To update a reader’s information, an *UPDATE* function will be used on the Readers table to update the reader's information with the condition that the username must equal the reader’s username. If updating the reader’s username, this will also include an *UPDATE* on the WritesReview and ReadsBook tables with the condition that the usernames match to update the username across all tables.
- *Updating Authors:* To update an author’s information, an *UPDATE* function will be used on the Authors table to update the author’s information with the condition that the username must equal the author’s username. If updating the author’s username, this will also include an *UPDATE* on the WritesBook table with the condition that the usernames match to update the username across all tables.
- *Updating Reviews:* To update a review, an *UPDATE* function will be used on the Reviews table to update the review information with the condition that the review IDs match.
- *Updating Reading Status:* To update the reading status of a book, an *UPDATE* function will be used with the conditions that the usernames and the ISBNs match.
- *Deleting a Book:* To delete a book from the Book Nook database, a *DELETE* will be used on BookAuthor, BookDetails, WritesBook, ReadsBook, Reviews, andWritesReview with the conditions that the ISBN is the same between the first five tables, the review ID is the same between Reviews and WritesReview, and the ISBN matches the book being deleted. This ensures all instance of the book are removed.
- *Deleting a Reader:* To delete a reader, a *DELETE* will be used on the Readers, ReadsBook, Review, and WritesReview tables with the conditions being that the username matches for the Readers, ReadsBook, and WritesReview table, the review IDs match for WritesReview and Reviews, and the username matches the specific reader being deleted. This ensures that all instances of the reader are removed.
- *Deleting an Author:* To delete an author, a *DELETE* will be used on the Author, and WritesBook tables with the conditions being that the username matches for both tables and that the username matches the specific author being deleted. This ensures that all instances of the author are removed. The books they have already entered will *not* be deleted.
- *Deleting a Review:* To delete a review, a *DELETE* will be used on Reviews and WritesReview with the conditions being that the review IDs are the same between both tables and that the review ID matches the specific review being deleted.

- *Removing Reading Status:* To remove the reading status from a book, a *DELETE* function will be used on the ReadsBook table with the condition that the username and ISBN match.
- *Displaying Author's Books:* To display all of a particular author's books, a *SELECT* will be used on the BookAuthor, BookDetails, and WritesBook tables connected with a *NATURAL JOIN* with the condition that the author's usernames match.
- *Displaying a Reader's Books:* To display the library of the reader, a *SELECT* will be used on the BookAuthor, BookDetails, and ReadsBook tables connected with a *NATURAL JOIN* with the condition that the reading status matches what the reader is looking for and that the username matches the reader.
- *Displaying Author's Reviews:* To display the review breakdown for the authors, a *SELECT* will be used on the Reviews table with the condition that the ISBN matches.
- *Displaying Reader's Reviews:* To display the review breakdown for the readers, a *NATURAL JOIN* will be completed on the Reviews, WritesReview, BookAuthor, and BookDetails tables. A *SELECT* will then be used with the conditions that the usernames match and the dates finished are between the reader's selected dates.

4.3 Operations Testing / Demonstration

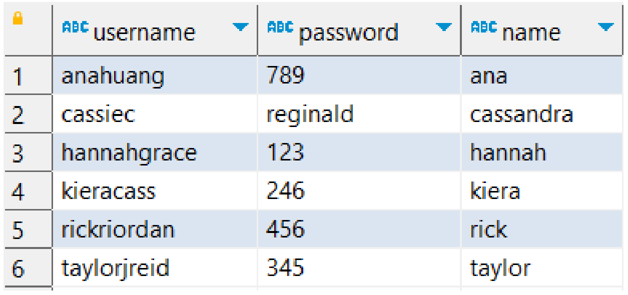
Inserting a Reader: As seen below, a reader was inserted into the Readers table of Figure 14.



	username	password	name
1	leishacraven	strawberry	leisha
2	lylyb	delta122	lynn
3	maeganlucas	meg	maegan
4	mandalynchance	ollie	mandalyn
5	sallyrooney2002	3005	esh

Figure 22: Result of inserting a new reader.

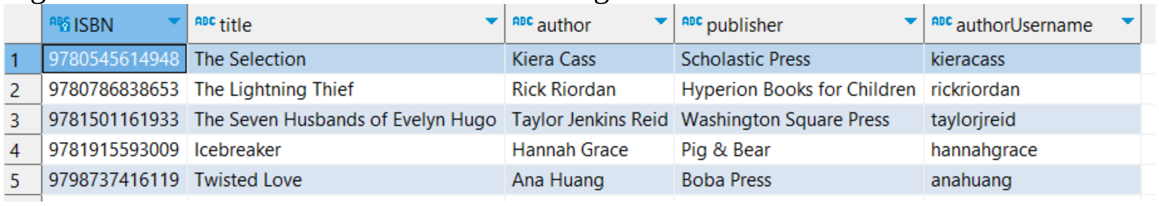
Inserting an Author: As seen below, an author was inserted into the Authors table of Figure 15.



	username	password	name
1	anahuang	789	ana
2	cassiec	reginald	cassandra
3	hannahgrace	123	hannah
4	kieracass	246	kiera
5	rickriordan	456	rick
6	taylorjreid	345	taylor

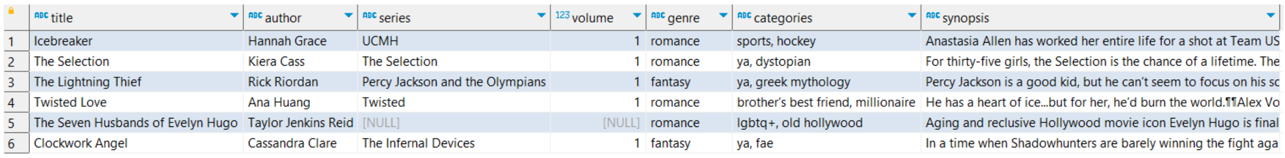
Figure 23: Result of inserting a new author.

Inserting a Book: As seen below, a book entry was made into the BookAuthor table of Figure 16 and into the BookDetails table of Figure 17.



	ISBN	title	author	publisher	authorUsername
1	9780545614948	The Selection	Kiera Cass	Scholastic Press	kieracass
2	9780786838653	The Lightning Thief	Rick Riordan	Hyperion Books for Children	rickriordan
3	9781501161933	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	Washington Square Press	taylorjreid
4	9781915593009	Icebreaker	Hannah Grace	Pig & Bear	hannahgrace
5	9798737416119	Twisted Love	Ana Huang	Boba Press	anahuang

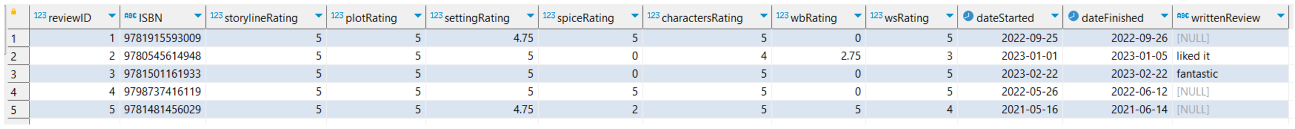
Figure 24: Result of inserting a new book's title, author, and publisher details.



	title	author	series	volume	genre	categories	synopsis
1	Icebreaker	Hannah Grace	UCMH	1	romance	sports, hockey	Anastasia Allen has worked her entire life for a shot at Team US
2	The Selection	Kiera Cass	The Selection	1	romance	ya, dystopian	For thirty-five girls, the Selection is the chance of a lifetime. The
3	The Lightning Thief	Rick Riordan	Percy Jackson and the Olympians	1	fantasy	ya, greek mythology	Percy Jackson is a good kid, but he can't seem to focus on his sc
4	Twisted Love	Ana Huang	Twisted	1	romance	brother's best friend, millionaire	He has a heart of ice...but for her, he'd burn the world.¶¶Alex Vo
5	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	[NULL]	[NULL]	romance	lgbtq+, old hollywood	Aging and reclusive Hollywood movie icon Evelyn Hugo is final
6	Clockwork Angel	Cassandra Clare	The Infernal Devices	1	fantasy	ya, fae	In a time when Shadowhunters are barely winning the fight aga

Figure 25: Result of inserting a new book's details.

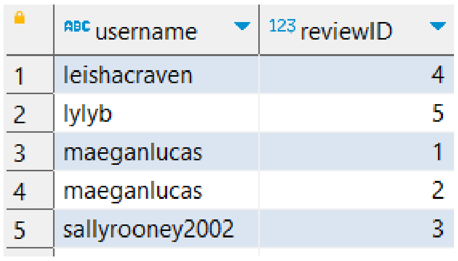
Inserting Reviews: As seen below, a review was inserted into the Reviews table of Figure 18.



	reviewID	ISBN	storylineRating	plotRating	settingRating	spiceRating	charactersRating	wbRating	wsRating	dateStarted	dateFinished	writtenReview
1	1	9781915593009	5	5	4.75	5	5	0	5	2022-09-25	2022-09-26	[NULL]
2	2	9780545614948	5	5	5	0	4	2.75	3	2023-01-01	2023-01-05	liked it
3	3	9781501161933	5	5	5	0	5	0	5	2023-02-22	2023-02-22	fantastic
4	4	9798737416119	5	5	5	5	5	0	5	2022-05-26	2022-06-12	[NULL]
5	5	9781481456029	5	5	4.75	2	5	5	4	2021-05-16	2021-06-14	[NULL]

Figure 26: Result of inserting a review.

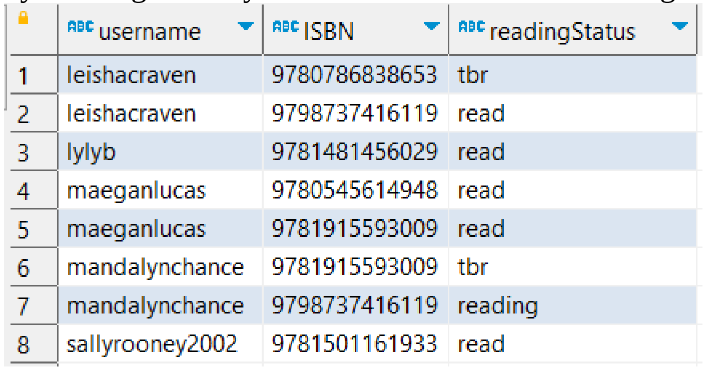
Connecting Reviews to Readers: As seen below, a review was connected to a reader by creating an entry in the WritesReview table of Figure 20:



	username	reviewID
1	leishacraven	4
2	lylyb	5
3	maeganlucas	1
4	maeganlucas	2
5	sallyrooney2002	3

Figure 27: Result of inserting an entry into the WritesReview table.

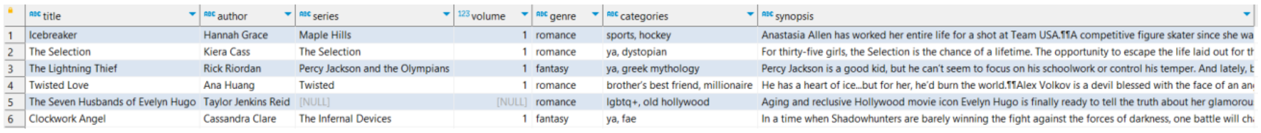
Choosing Reading Status: As seen below, a reader was able to choose their reading status by creating an entry into the ReadsBook table of Figure 19.



	username	ISBN	readingStatus
1	leishacraven	9780786838653	tbr
2	leishacraven	9798737416119	read
3	lylyb	9781481456029	read
4	maeganlucas	9780545614948	read
5	maeganlucas	9781915593009	read
6	mandalynchance	9781915593009	tbr
7	mandalynchance	9798737416119	reading
8	sallyrooney2002	9781501161933	read

Figure 29: The result of adding a reading status to the ReadsBook table.

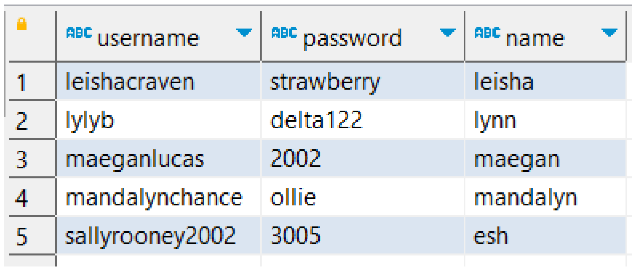
Updating Book Information: As seen below, the series *Icebreaker* was updated from that of Figure 25.



	title	author	series	volume	genre	categories	synopsis
1	Icebreaker	Hannah Grace	Maple Hills	1	romance	sports, hockey	Anastasia Allen has worked her entire life for a shot at Team USA.¶¶A competitive figure skater since she wa
2	The Selection	Kiera Cass	The Selection	1	romance	ya, dystopian	For thirty-five girls, the Selection is the chance of a lifetime. The opportunity to escape the life laid out for th
3	The Lightning Thief	Rick Riordan	Percy Jackson and the Olympians	1	fantasy	ya, greek mythology	Percy Jackson is a good kid, but he can't seem to focus on his schoolwork or control his temper. And lately, b
4	Twisted Love	Ana Huang	Twisted	1	romance	brother's best friend, millionaire	He has a heart of ice...but for her, he'd burn the world.¶¶Alex Volkov is a devil blessed with the face of an an
5	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	[NULL]	[NULL]	romance	lgbtq+, old hollywood	Aging and reclusive Hollywood movie icon Evelyn Hugo is finally ready to tell the truth about her glamorou
6	Clockwork Angel	Cassandra Clare	The Infernal Devices	1	fantasy	ya, fae	In a time when Shadowhunters are barely winning the fight against the forces of darkness, one battle will ch

Figure 30: Result of updating book information.

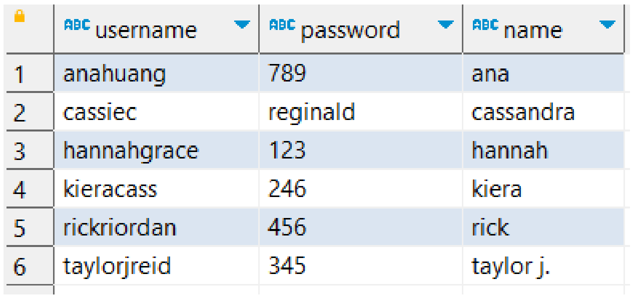
Updating Readers: As seen below, the password changed from "meg" in Figure 22 to "2002".



	username	password	name
1	leishacraven	strawberry	leisha
2	lylyb	delta122	lynn
3	maeganlucas	2002	maegan
4	mandalynchance	ollie	mandalyn
5	sallyrooney2002	3005	esh

Figure 31: Result of changing a reader's password.

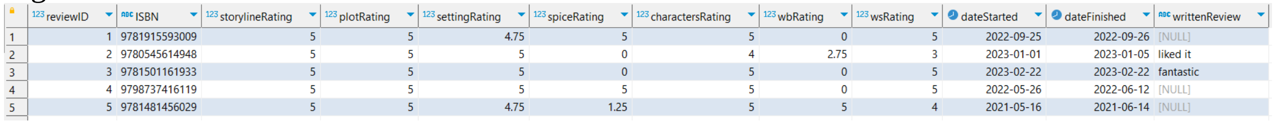
Updating Authors: As seen below, the name changed from "taylor" in Figure 23 to "taylor j.".



	username	password	name
1	anahuang	789	ana
2	cassiec	reginald	cassandra
3	hannahgrace	123	hannah
4	kieracass	246	kiera
5	rickriordan	456	rick
6	taylorjreid	345	taylor j.

Figure 32: Result of changing an author's name.

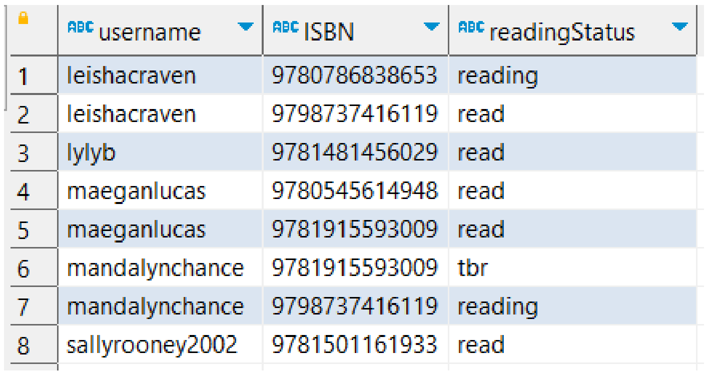
Updating Reviews: As seen below, the spice rating for review 5 has changed from 2 in Figure 26 to 1.25



	reviewID	ISBN	storylineRating	plotRating	settingRating	spiceRating	charactersRating	wbRating	wsRating	dateStarted	dateFinished	writtenReview
1	1	9781915593009	5	5	4.75	5	5	0	5	2022-09-25	2022-09-26	[NULL]
2	2	9780545614948	5	5	5	0	4	2.75	3	2023-01-01	2023-01-05	liked it
3	3	9781501161933	5	5	5	0	5	0	5	2023-02-22	2023-02-22	fantastic
4	4	9798737416119	5	5	5	5	5	0	5	2022-05-26	2022-06-12	[NULL]
5	5	9781481456029	5	5	4.75	1.25	5	5	4	2021-05-16	2021-06-14	[NULL]

Figure 33: Result of changing the spice rating for review number 5.

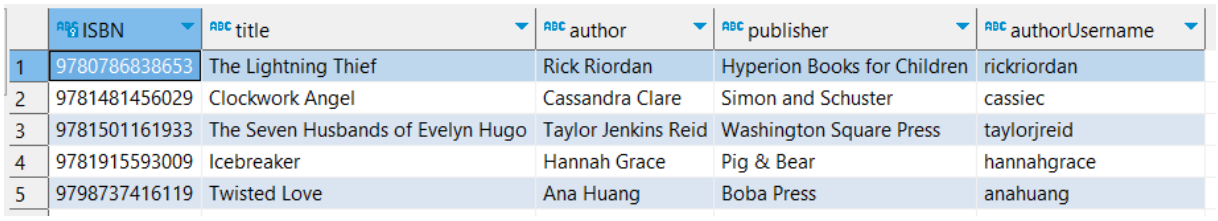
Updating Reading Status: As seen below, the reading status changed from “tbr” to “reading” for the first entry of Figure 29.



	username	ISBN	readingStatus
1	leishacraven	9780786838653	reading
2	leishacraven	9798737416119	read
3	lylyb	9781481456029	read
4	maeganlucas	9780545614948	read
5	maeganlucas	9781915593009	read
6	mandalynchance	9781915593009	tbr
7	mandalynchance	9798737416119	reading
8	sallyrooney2002	9781501161933	read

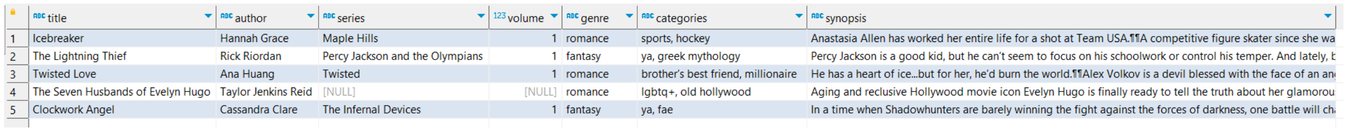
Figure 34: Results of changing a reading status.

Deleting a Book: As seen below the book *The Selection* and all instances of its ISBN, 9780545614948, was deleted from all tables.



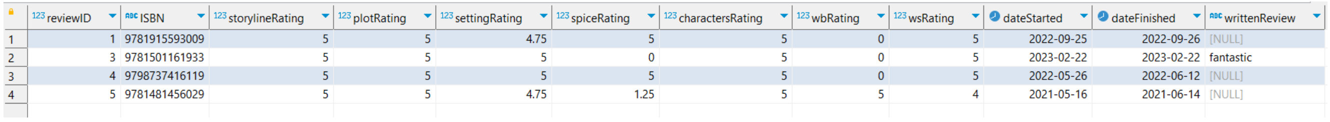
	ISBN	title	author	publisher	authorUsername
1	9780786838653	The Lightning Thief	Rick Riordan	Hyperion Books for Children	rickriordan
2	9781481456029	Clockwork Angel	Cassandra Clare	Simon and Schuster	cassiec
3	9781501161933	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	Washington Square Press	taylorjreid
4	9781915593009	Icebreaker	Hannah Grace	Pig & Bear	hannahgrace
5	9798737416119	Twisted Love	Ana Huang	Boba Press	anahuang

Figure 35: Result of deleting a book from the BookAuthor table.



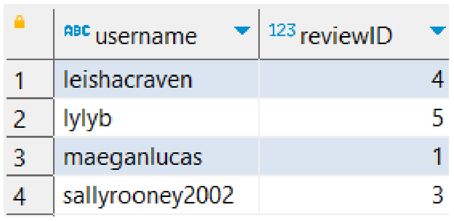
	title	author	series	volume	genre	categories	synopsis
1	Icebreaker	Hannah Grace	Maple Hills	1	romance	sports, hockey	Anastasia Allen has worked her entire life for a shot at Team USA.¶¶A competitive figure skater since she wa
2	The Lightning Thief	Rick Riordan	Percy Jackson and the Olympians	1	fantasy	ya, greek mythology	Percy Jackson is a good kid, but he can't seem to focus on his schoolwork or control his temper. And lately, b
3	Twisted Love	Ana Huang	Twisted	1	romance	brother's best friend, millionaire	He has a heart of ice...but for her, he'd burn the world.¶¶Alex Volkov is a devil blessed with the face of an an
4	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	[NULL]	[NULL]	romance	lgbtq+, old hollywood	Aging and reclusive Hollywood movie icon Evelyn Hugo is finally ready to tell the truth about her glamorou
5	Clockwork Angel	Cassandra Clare	The Infernal Devices	1	fantasy	ya, fae	In a time when Shadowhunters are barely winning the fight against the forces of darkness, one battle will ch

Figure 36: Result of deleting a book from the BookDetails table.



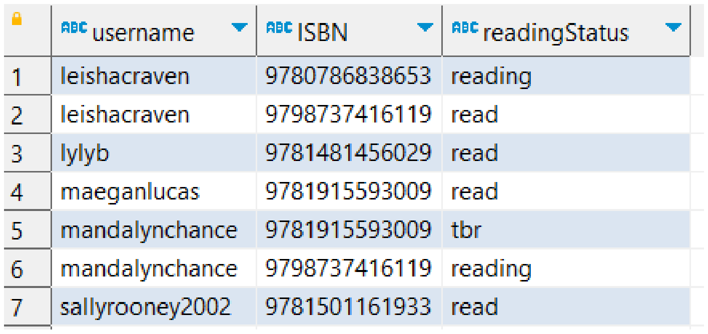
	reviewID	ISBN	storylineRating	plotRating	settingRating	spiceRating	charactersRating	wbRating	wsRating	dateStarted	dateFinished	writtenReview
1	1	9781915593009	5	5	4.75	5	5	0	5	2022-09-25	2022-09-26	[NULL]
2	3	9781501161933	5	5	5	0	5	0	5	2023-02-22	2023-02-22	fantastic
3	4	9798737416119	5	5	5	5	5	0	5	2022-05-26	2022-06-12	[NULL]
4	5	9781481456029	5	5	4.75	1.25	5	5	4	2021-05-16	2021-06-14	[NULL]

Figure 37: Results of deleting a book from the Reviews table.



	username	reviewID
1	leishacraven	4
2	lylyb	5
3	maeganlucas	1
4	sallyrooney2002	3

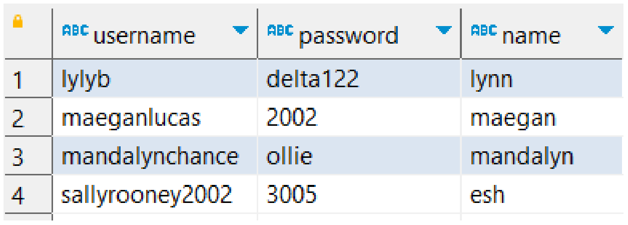
Figure 38: Results of deleting an associated book from the WritesReview table.



	username	ISBN	readingStatus
1	leishacraven	9780786838653	reading
2	leishacraven	9798737416119	read
3	lylyb	9781481456029	read
4	maeganlucas	9781915593009	read
5	mandalynchance	9781915593009	tbr
6	mandalynchance	9798737416119	reading
7	sallyrooney2002	9781501161933	read

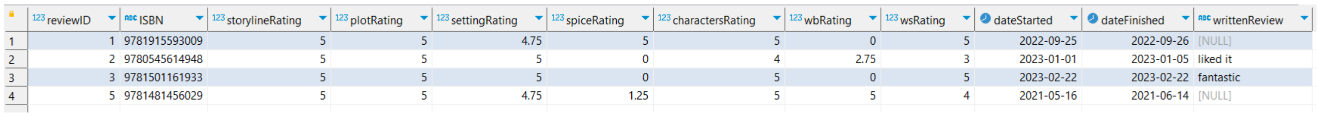
Figure 40: Results of deleting a book from the ReadsBook table.

Deleting a Reader: As seen below, all instances of the reader "leishacraven" have been removed from all the associated tables.



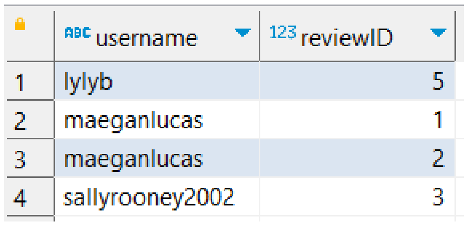
	username	password	name
1	lylyb	delta122	lynn
2	maeganlucas	2002	maegan
3	mandalynchance	ollie	mandalyn
4	sallyrooney2002	3005	esh

Figure 41: Results of deleting a reader from the Readers table.



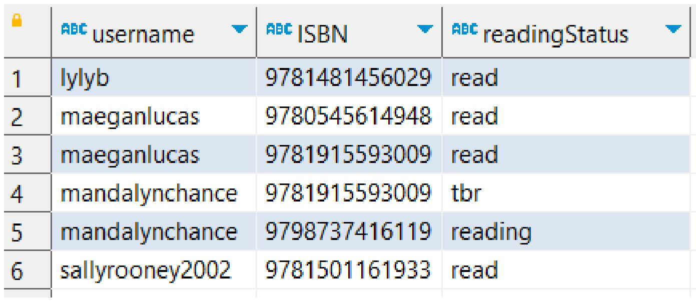
	reviewID	ISBN	storylineRating	plotRating	settingRating	spiceRating	charactersRating	wbRating	wsRating	dateStarted	dateFinished	writtenReview
1	1	9781915593009	5	5	4.75	5	5	0	5	2022-09-25	2022-09-26	[NULL]
2	2	9780545614948	5	5	5	0	4	2.75	3	2023-01-01	2023-01-05	liked it
3	3	9781501161933	5	5	5	0	5	0	5	2023-02-22	2023-02-22	fantastic
4	5	9781481456029	5	5	4.75	1.25	5	5	4	2021-05-16	2021-06-14	[NULL]

Figure 42: Results of deleting an associated reader from the Reviews table.



	username	reviewID
1	lylyb	5
2	maeganlucas	1
3	maeganlucas	2
4	sallyrooney2002	3

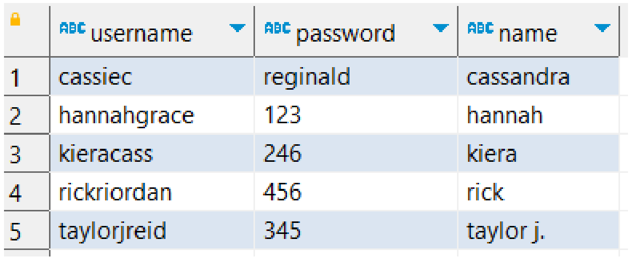
Figure 43: Results of deleting a reader from the WritesReview table.



	username	ISBN	readingStatus
1	lylyb	9781481456029	read
2	maeganlucas	9780545614948	read
3	maeganlucas	9781915593009	read
4	mandalynchance	9781915593009	tbr
5	mandalynchance	9798737416119	reading
6	sallyrooney2002	9781501161933	read

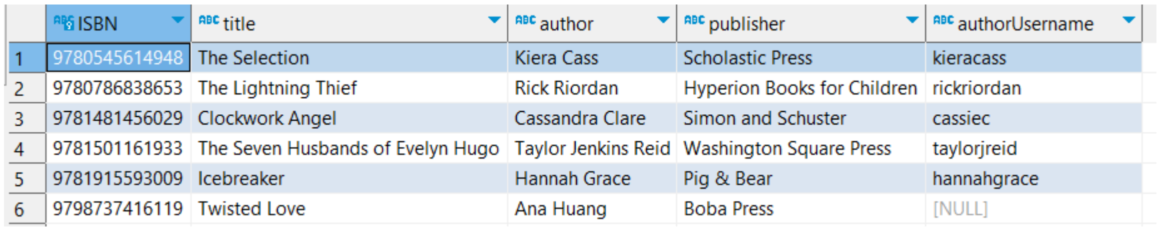
Figure 44: Results of deleting a reader from the ReadsBook table.

Deleting an Author: As seen below, all instances of the author "anahuang" have been removed from all associated tables.



	username	password	name
1	cassiec	reginald	cassandra
2	hannahgrace	123	hannah
3	kieracass	246	kiera
4	rickriordan	456	rick
5	taylorjreid	345	taylor j.

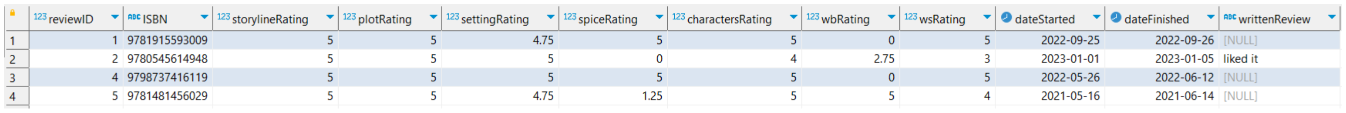
Figure 45: Results of deleting an author from the Authors table.



	ISBN	title	author	publisher	authorUsername
1	9780545614948	The Selection	Kiera Cass	Scholastic Press	kieracass
2	9780786838653	The Lightning Thief	Rick Riordan	Hyperion Books for Children	rickriordan
3	9781481456029	Clockwork Angel	Cassandra Clare	Simon and Schuster	cassiec
4	9781501161933	The Seven Husbands of Evelyn Hugo	Taylor Jenkins Reid	Washington Square Press	taylorjreid
5	9781915593009	Icebreaker	Hannah Grace	Pig & Bear	hannahgrace
6	9798737416119	Twisted Love	Ana Huang	Boba Press	[NULL]

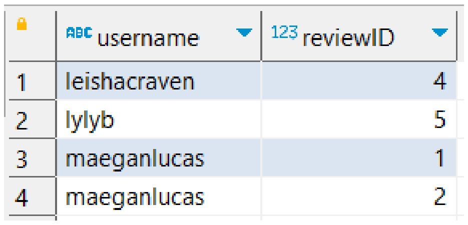
Figure 46: Results of deleting an author from the BookAuthor table.

Deleting a Review: As seen below, all instances of the review with review ID number 3 have been deleted from the associated tables.



	reviewID	ISBN	storylineRating	plotRating	settingRating	spiceRating	charactersRating	wbRating	wsRating	dateStarted	dateFinished	writtenReview
1	1	9781915593009	5	5	4.75	5	5	0	5	2022-09-25	2022-09-26	[NULL]
2	2	9780545614948	5	5	5	0	4	2.75	3	2023-01-01	2023-01-05	liked it
3	4	9798737416119	5	5	5	5	5	0	5	2022-05-26	2022-06-12	[NULL]
4	5	9781481456029	5	5	4.75	1.25	5	5	4	2021-05-16	2021-06-14	[NULL]

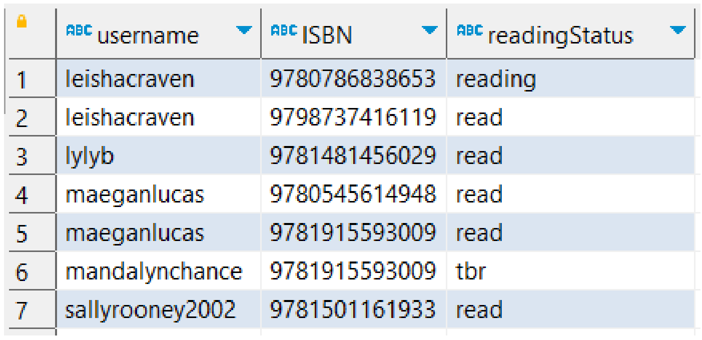
Figure 47: Results of deleting a review from the Reviews table.



	username	reviewID
1	leishacraven	4
2	lylyb	5
3	maeganlucas	1
4	maeganlucas	2

Figure 48: Results of deleting a review from the WritesReview table.

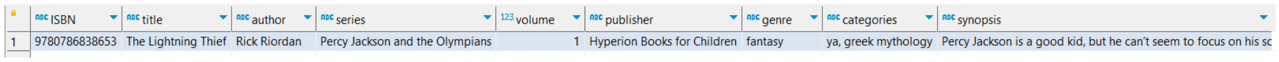
Removing a Reading Status: As seen below, one of the reading statuses for the reader "mandalynchance" was deleted from Figure 29.



	username	ISBN	readingStatus
1	leishacraven	9780786838653	reading
2	leishacraven	9798737416119	read
3	lylyb	9781481456029	read
4	maeganlucas	9780545614948	read
5	maeganlucas	9781915593009	read
6	mandalynchance	9781915593009	tbr
7	sallyrooney2002	9781501161933	read

Figure 49: Results of deleting a reading status from the ReadsBook table.

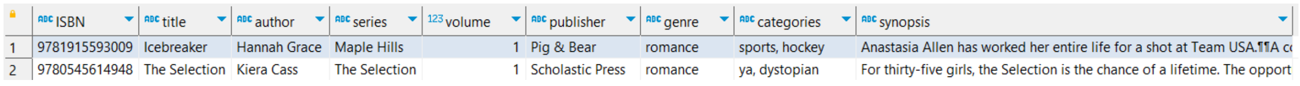
Displaying an Author's Books: Below is the result of the query for books written by author "rickriordan".



	ISBN	title	author	series	volume	publisher	genre	categories	synopsis
1	9780786838653	The Lightning Thief	Rick Riordan	Percy Jackson and the Olympians	1	Hyperion Books for Children	fantasy	ya, greek mythology	Percy Jackson is a good kid, but he can't seem to focus on his sc

Figure 50: Results of displaying a specific author's books.

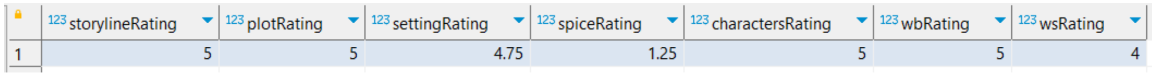
Displaying a Reader's Books: Below is the result of the query for the books of the reader "maeganlucas" with a reading status of "read". The information displayed will be all the attributes in both the BookAuthor and BookDetails tables.



	ISBN	title	author	series	volume	publisher	genre	categories	synopsis
1	9781915593009	Icebreaker	Hannah Grace	Maple Hills	1	Pig & Bear	romance	sports, hockey	Anastasia Allen has worked her entire life for a shot at Team USA.¶¶A c
2	9780545614948	The Selection	Kiera Cass	The Selection	1	Scholastic Press	romance	ya, dystopian	For thirty-five girls, the Selection is the chance of a lifetime. The opport

Figure 51: Results of displaying a specific reader's read books.

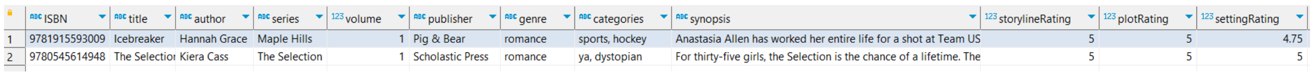
Displaying Author's Reviews: Below is the result of the query for reviews written about the book with ISBN 9781481456029.



	storylineRating	plotRating	settingRating	spiceRating	charactersRating	wbRating	wsRating
1	5	5	4.75	1.25	5	5	4

Figure 52: Results of displaying ratings for a specific book to its author.

Displaying Reader's Reviews: Below is the result of the query for reviews written by reader "maeganlucas" between the selected dates of 06/01/2022 and 06/01/2023 as well as the books' details.



	ISBN	title	author	series	volume	publisher	genre	categories	synopsis	storylineRating	plotRating	settingRating
1	9781915593009	Icebreaker	Hannah Grace	Maple Hills	1	Pig & Bear	romance	sports, hockey	Anastasia Allen has worked her entire life for a shot at Team US	5	5	4.75
2	9780545614948	The Selectioı	Kiera Cass	The Selection	1	Scholastic Press	romance	ya, dystopian	For thirty-five girls, the Selection is the chance of a lifetime. The	5	5	5

Figure 53: Results of displaying rating and book details for a specific reader in a specified date range.

5 Deliverable 4: Application Implementation

This section briefly describes the application that was developed. Appendix A contains a code listing of all applications. In this section, each source file is briefly described in the same order in which it is presented in the appendix.

A video demonstration of the application can be found here. You'll notice that the read books lists three books, but the review screen says that there were 4 books read – this is because the book *Icebreaker* was read twice in the given time frame.

5.1 AuthorHomeScreen.java

Description	This file is the main home screen for an Author user. It sets the menu bar and allows interaction that changes the center of the screen. It is the main hub for an Author user. This screen is automatically displayed when an Author user logs in.
User Input	User input is taken in the form of mouse clicks on the menu bar. Clicking *Create Book* places the new book pane for the author to use. Clicking *Update Book* places the update book pane for the author to use. Clicking *Delete Book* places the delete book popup for the author to use. Clicking *See Reviews* takes them to the review screen. Clicking *Back to Home* brings them back to the home page. Clicking *Logout* logs the user out and brings the application back to the log in screen. Clicking *Update Profile* brings up the author profile update screen. Clicking *Delete Profile* brings up the delete confirmation popup.
User Output	The menu bar is shown to the user which contains the sections *Books*, *Reviews*, *Settings*, and *Logout*. The *Books* menu contains *Create Book*, *Update Book*, and *Delete Book*. The *Reviews* menu contains *See Reviews*. The *Settings* menu contains *Update Profile* and *Delete Profile*. The *Logout* menu contains *Back to Home* and *Logout*.
Database Operations	No database operations are performed by this file.

5.2 AuthorProfileUpdate.java

Description	This file allows an Author user to update their profile information. It will be placed in the center of the author's home screen. The file interacts with other files to allow for the update of the profile information within the database. This file is displayed when the *Update Profile* menu item is selected from the author's home screen.
User Input	This file takes user input from text fields. The *name* text field will take the input to be used as the author's preferred name. The *username* text field will take the input to be used as the author's username. The *password* text field will take the input to be used as

	the author’s password. The button *update* will take a mouse click input, and upon this action, will go back to the author home screen and indicate to the *MainUI* file to initiate the database operation.
User Output	This file shows the *name*, *username*, and *password* text fields and the *update* button. It also displays the labels to indicate which field is which and the instructions. The text fields will be filled with the author’s profile information.
Database Operations	No database operations are performed by this file.

5.3 AuthorReviewPane.java

Description	This file allows for an Author user to choose from their books within the database to see its reviews. This file is displayed in the center of the author’s home screen. Once a book is selected, it interacts with the *MainUI* file to place charts for each review category in the middle of this file. This file is displayed when the *See Reviews* menu item is selected in the author’s home screen.
User Input	User input is taken in the form of a combo box, or drop down menu. This input will be the book that is used to pull reviews from the database. Input will also be taken from a button to see the reviews for the selected book.
User Output	A label with instructions is displayed to the user. A combo box, *book*, with the books written by the author that is currently in the database is displayed. It allows the user to select which book to see reviews for. The button *seeReviews* is displayed to the user, and when clicked, will interact with the *MainGUI* file to pull the reviews for the specific book.
Database Operations	Database operations to select and display the author’s books are called upon by this file, but are not performed by this file.

5.4 BookCreationPane.java

Description	This file allows for an Author user to create a new book which will be entered into the database with interactions with other files. This pane is displayed in the center of the author’s home screen. The file is displayed when the *Create Book* menu item is selected from the author’s home screen.
User Input	User input is taken in the form of multiple text fields: *title*, *author*, *series*, *volume*, *ISBN*, *publisher*, and *categories*. These become the respective attributes of the new book. A combo box, *genre*, takes user input that becomes the genre attribute of the new book. A text area, *synopsis*, takes user input to become the synopsis attribute of the book. The button *createBook* takes mouse input and when clicked will indicate to the necessary files to insert the values into the database.

User Output	The user input fields *title*, *author*, *series*, *volume*, *ISBN*, *publisher*, *categories*, *genre*, and *synopsis* are displayed to the user. The button *createBook* is displayed to the user. Labels for each input field are also displayed to the user.
Database Operations	No database operations are performed by this file.

5.5 BookSearchPane.java

Description	This file allows for a Reader to search the books that have been entered by authors into the database. It is displayed in the center of the reader's home screen. The input from this file will be used to search the database of books. Input from this file this pane is displayed when the *Search Book* menu item is selected from the reader's home screen.
User Input	User input is taken in the form of a text field that acts as a search bar. A button, *search,* is used to signal when the other files should pull the search results from the database.
User Output	A label with instructions, the text field *searchBar,* and the button *search* are displayed to the user at the top of the pane. Once the button is pressed and the search results are pulled, another pane will be added to the center containing the search results and buttons to set the reading status.
Database Operations	No database operations are performed by this file.

5.6 BookUpdatePane.java

Description	This file allows for an Author to update the information of a book they have entered into the database. This pane is displayed in the center of the author's home screen. The user input from this file will become the specific book's new attributes. This file is displayed when the *Update Book* menu item is selected in the author's home screen.
User Input	User input is taken in the form of multiple text fields: *title, author, series*, *volume*, *ISBN*, *publisher,* and *categories*. These become the respective attributes of the new book. A combo box, *genre*, takes user input that becomes the genre attribute of the new book. A text area, *synopsis*, takes user input to become the synopsis attribute of the book. The button *updateBook* takes mouse input and when clicked will indicate to the necessary files to update the values into the database.
User Output	The user input fields *title*, *author*, *series*, *volume*, *ISBN*, *publisher*, *categories*, *genre*, and *synopsis* are displayed to the user. The button *createBook* is displayed to the user. Labels for each input field are

	also displayed to the user. The input fields are filled with the book's details from the database.
Database Operations	The database operation of selecting the selected book's details are called upon by this file, but are not actually performed by this file.

5.7 LoginScreen.java

Description	This file allows for both Author and Reader users to log in to the application. The input is checked against the database to make sure it is a valid user combination. This login information is used to determine which home screen and functionalities to display to the user. This screen is automatically displayed upon opening the application.
User Input	User input is taken in the form of text fields, *username* and *password*. A button, *login*, will take mouse input and when clicked will indicate to the necessary files to verify the information in the database. A button, *newUser*, will take mouse input and when clicked will display the registration panes
User Output	The user input fields *username* and *password* as well as the buttons *login* and *newUser* are displayed to the user. A picture of the Book Nook logo is displayed to the user as well. An alert is shown to the user if the login credentials are unable to be verified.
Database Operations	No database operations are performed by this file.

5.8 MainUI.java

Description	This file is the main file for user interactions and calling upon database interactions. All event handling is completed in this file as well as all pane changes. This file interacts with or is interacted with by all other files. This file creates panes within the file that are used in different capacities. This files also creates popups that allow for user input for different functionalities. This file also creates the charts for the review screens of both types of users.
User Input	User input is taken from text fields in popups that are used to update and delete books and reviews. The input from these popups are the ISBN attributes of books that will be used to pull the proper information from the database. Buttons are also used to take mouse input that will signal the database to delete or pull from the database. Text fields and text areas are used to take input for reviews.
User Output	The text fields, popups, and buttons for user input to create a review, update a review, delete or update a book, or delete a profile are displayed to the user. Multiple labels are used to display the author's books, the reader's books of all three statuses, and the search results of a reader. There are alerts that are shown to the user upon the

	deletion of the profile and error alerts shown to the user when an invalid ISBN is entered.
Database Operations	Database operations to insert, select, update, and delete from all database tables are called upon by this file, but are not actually performed by this file.

5.9 Manager.java

Description	This file is used for database operations only. All database operations are performed by this file. This functions in this file are called upon by multiple other files to create, select, update, or delete information from the database.
User Input	There is no user input taken by this file.
User Output	There is no output shown to the user from this file.
Database Operations	All database operations are completed by this file. Inserting book details, authors, readers, reviews, reading statuses, and connections between reader and review are completed is this file. Selecting an author's books, a reader's books based on the reading status, the reviews of a book, the reviews of a reader, the profile information of both author and reader, and the names of both author and reader is completed by this file. The deletion of books, profiles, reviews, and reading statuses is completed by this file. Updating book information, profile information, review information, or reading statuses is completed by this file.

5.10 ReaderHomeScreen.java

Description	This file is the main home screen for a Reader user. It sets the menu bar and allows interaction that changes the center of the screen. It is the main hub for Reader users. This screen is automatically displayed when a Reader logs in.
User Input	User input is taken in the form of mouse clicks on the menu bar. Clicking *Search Book* places the pane for searching the database of books in the center. Clicking *Read Books* displays the books that the reader has marked as "Read" in the center of the home screen. Clicking *Currently Reading* displays the books that the reader has marked "Currently Reading" in the center of the home screen. Clicking *TBR* displays the books that the reader has marked as "Want to Read" in the center of the home screen. Clicking *Update Reviews* brings up the popup that allows inputting the ISBN of the book to update its review. Clicking *Delete Reviews* brings up the popup that allows inputting the ISBN of the book to delete its review. Clicking *See Reviews* brings up the pane to select the dates of reviews to pull from the database. Clicking *Update Profile* brings up the reader profile update screen. Clicking *Delete Profile* brings up the delete

	confirmation popup. Clicking *Logout* logs the user out and brings the application back to the log in screen.
User Output	The menu bar is shown to the user which contains the sections *Books*, *Reviews*, *Settings*, and *Logout*. The *Books* menu contains *See Books*, *Read Books*, *Currently Reading*, and *TBR*. The *Reviews* menu contains *Update Review*, *Delete Review*, and *See Reviews*. The *Settings* menu contains *Update Profile* and *Delete Profile*. The *Logout* menu contains *Logout*.
Database Operations	No database operations are performed by this file.

5.11 ReaderProfileUpdate.java

Description	This file allows Reader to update their profile information. It will be placed in the center of the reader's home screen. The file interacts with other files to allow for the update of the profile information within the database. This file is displayed when the *Update Profile* menu item is selected from the reader's home screen.
User Input	This file takes user input from text fields. The *name* text field will take the input to be used as the reader's preferred name. The *username* text field will take the input to be used as the reader's username. The *password* text field will take the input to be used as the reader's password. The button *update* will take a mouse click input, and upon this action, will go back to the reader home screen and indicate to the *MainUI* file to initiate the database operation.
User Output	This file shows the *name*, *username*, and *password* text fields and the *update* button. It also displays the labels to indicate which field is which and the instructions. The text fields will be filled with the reader's profile information.
Database Operations	No database operations are performed by this file.

5.12 ReaderReviewPane.java

Description	This file allows for a Reader to choose from the dates between which to pull reviews from the database. This file is displayed in the center of the reader's home screen. Once the dates are selected, it interacts with the *MainUI* file to place charts for each review category in the middle of this file. This file is displayed when the *See Reviews* menu item is selected in the reader's home screen.
User Input	This file takes user input from text fields for the start date's month, day, and year as well as the finish date's month, day, and year. There is a button, *getReviews*, that will take mouse input to signal to the necessary files to get the reviews and display the charts.
User Output	This file shows text fields for the start date's month, day, and year as well as the finish date's month, day, and year. It also displays the

	getReviews button. It also displays the labels to indicate which field is which and the instructions.
Database Operations	No database operations are performed by this file.

5.13 ReadingStatusPane.java

Description	This file allows for a Reader to set the reading status for a book. Once a reader searches for a book, they choose a button to set a reading status that displays this file in the center of the reader's home screen. Once the book is pulled, multiple buttons will allow for the user to select what to do with the reading status information.
User Input	This file takes user input from a text field with the ISBN that will be prefilled when selecting a book from the search pane. Buttons called *search*, *setReadingStatus*, *editReadingStatus*, and *removeReadingStatus* that will take mouse input and interact with the necessary files to initiate the appropriate database operation. A combo box with the reading statuses will be used for the reader to set their reading status. It will be preset if there is already a status for that book.
User Output	This file shows a text field with the prefilled ISBN from selecting a book from the search pane. The buttons *search*, *setReadingStatus*, *editReadingStatus*, and *removeReadingStatus* will be shown to the reader. The combo box with the reading statuses will be shown to the user and if a particular book has an existing reading status, the combo box will be preset.
Database Operations	No database operations are performed by this file.

5.14 RegistrationPane.java

Description	This file allows for a new user to register and create a profile in Book Nook. This pane will be shown when the user selects to create a new profile. Once the user registers, the file will interact with the necessary files to create a user of a certain role in the database and returns to the login pane.
User Input	This file takes user input from text fields for a name, username, and password. A checkbox for both an author and a reader to determine the role of the user. A button takes mouse inputs to signal the necessary files to create a user in the database.
User Output	This file shows the text fields for the name, username, and password. The checkbox to determine the user's role and the button to create a user is displayed from this file. This file also displays labels for each user input and a welcome text display that also displays the Book Nook logo image.

Database Operations	No database operations are performed by this file.

6 Conclusion

The design for Book Nook has been completed. Mockups of the desired applications have been designed. A general basis of what information needs to be taken, stored, or received has been established. An Entity-Relationship Model has been made to reflect Book Nook and the relations needed for implementation have been determined based on this model. The database for Book Nook has been implemented, tested, and connected to an application. This application handles user interfacing for the database. Users can do all necessary operations from within the application. With this, the implementation of Book Nook is complete.

References

"39 using text in javafx," *39 Using Text in JavaFX (Release 8)*. [Online]. Available: https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/text-settings.htm. [Accessed: 27-Apr-2023].

A. Redko, "Release: Javafx 2.2," *Using JavaFX Charts: Pie Chart | JavaFX 2 Tutorials and Documentation*, 06-Dec-2013. [Online]. Available: https://docs.oracle.com/javafx/2/charts/pie-chart.htm#CIHFDADD. [Accessed: 27-Apr-2023].

A. Redko, "Release: Javafx 2.2," *Using JavaFX UI Controls: Combo Box | JavaFX 2 Tutorials and Documentation*, 27-Aug-2013. [Online]. Available: https://docs.oracle.com/javafx/2/ui_controls/combo-box.htm. [Accessed: 27-Apr-2023].

"Alert," *Alert (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Alert.html. [Accessed: 27-Apr-2023].

"CheckBox," *Checkbox (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/CheckBox.html. [Accessed: 27-Apr-2023].

"Color," *Color (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/paint/Color.html. [Accessed: 27-Apr-2023].

"Delete from two tables in one query," *Stack Overflow*, 01-May-1956. [Online]. Available: https://stackoverflow.com/questions/1233451/delete-from-two-tables-in-one-query. [Accessed: 02-Apr-2023].

"Font," *Font (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/text/Font.html. [Accessed: 27-Apr-2023].

"Get Screen Size," *Get screen size : Screen " javafx " java*. [Online]. Available: http://www.java2s.com/Code/Java/JavaFX/GetScreensize.htm. [Accessed: 27-Apr-2023].

"Image," *Image (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/image/Image.html. [Accessed: 27-Apr-2023].

"ImageView," *ImageView (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/image/ImageView.html. [Accessed: 27-Apr-2023].

J. D. Ullman and J. Widom, *A first course in database systems*. Upper Saddle River, NJ: Prentice Hall, 2002.

J. Gordon, "Release: Javafx 2.1," *Working With Layouts in JavaFX: Using Built-in Layout Panes | JavaFX 2 Tutorials and Documentation*, 24-Apr-2013. [Online]. Available: https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm. [Accessed: 27-Apr-2023].

J. Monsalve, F. Fabian, and G. Veres, "How to center a label on a pane in javafx," *Stack Overflow*, 01-Jan-1963. [Online]. Available: https://stackoverflow.com/questions/36854031/how-to-center-a-label-on-a-pane-in-javafx. [Accessed: 27-Apr-2023].

"Labeled," *Labeled (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Labeled.html. [Accessed: 27-Apr-2023].

"Pane," *Pane (javafx 2.2)*, 09-Dec-2013. [Online]. Available: https://docs.oracle.com/javafx/2/api/javafx/scene/layout/Pane.html. [Accessed: 27-Apr-2023].

"Popup," *Popup (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/stage/Popup.html. [Accessed: 27-Apr-2023].

R. Stansbury, "Files/Database Systems Canvas Course," *CS 317 - Files/Database Systems*. [Online]. Available: https://erau.instructure.com/courses/154254. [Accessed: 09-Feb-2023].

"TextAlignment," *Textalignment (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/text/TextAlignment.html. [Accessed: 27-Apr-2023].

"TextArea," *Textarea (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TextArea.html. [Accessed: 27-Apr-2023].

"TextField," *Textfield (javafx 8)*, 10-Feb-2015. [Online]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TextField.html. [Accessed: 27-Apr-2023].

user10484841 and S. Gusaroff, “Java.lang.illegalargumentexception: Invalid URL or resource NOT FOUND,” *Oracle Forums*, 07-Jun-2013. [Online]. Available: https://forums.oracle.com/ords/apexds/post/java-lang-illegalargumentexception-invalid-url-or-resource-1421. [Accessed: 27-Apr-2023].

Appendix A: Source File Listing

I. AuthorHomeScreen.java

```
import javafx.scene.control.Menu;
import javafx.scene.control.MenuBar;
import javafx.scene.control.MenuItem;
import javafx.scene.layout.BorderPane;

public class AuthorHomeScreen extends BorderPane {

    private MenuBar menuBar;
    private MenuItem menuCreate, menuUpdate, menuDelete,
menuReviews, menuLogout, menuBack, menuUpdateProfile,
menuDeleteProfile;

    public AuthorHomeScreen() {
        createAuthorHomeScreen();

    }

    public void createAuthorHomeScreen() {
        this.setStyle("-fx-background-color: antiquewhite;");
        menuBar = new MenuBar();
        Menu books = new Menu("Books");
        menuCreate = new MenuItem("Create Book");
        menuUpdate = new MenuItem("Update Book");
        menuDelete = new MenuItem("Delete Book");
        Menu reviews = new Menu("Reviews");
        menuReviews = new MenuItem("See Reviews");
        Menu logout = new Menu("Logout");
        menuBack = new MenuItem("Back to Home");
        menuLogout = new MenuItem("Logout");
        Menu settings = new Menu("Settings");
        menuUpdateProfile = new MenuItem("Update Profile");
        menuDeleteProfile = new MenuItem("Delete Profile");

        books.getItems().addAll(menuCreate, menuUpdate,
menuDelete);
        reviews.getItems().add(menuReviews);
        logout.getItems().addAll(menuBack, menuLogout);
        settings.getItems().addAll(menuUpdateProfile,
menuDeleteProfile);
        menuBar.getMenus().addAll(books, reviews, settings,
logout);
        this.setTop(menuBar);

    }

    public MenuItem getMenuCreate() {
        return menuCreate;
    }

    public MenuItem getMenuUpdate() {
        return menuUpdate;
    }
    public MenuItem getMenuDelete() {
        return menuDelete;
    }

    public MenuItem getMenuReviews() {
        return menuReviews;
    }

    public MenuItem getMenuUpdateProfile() {
        return menuUpdateProfile;
    }

    public MenuItem getMenuDeleteProfile() {
        return menuDeleteProfile;
    }

    public MenuItem getMenuLogout() {
        return menuLogout;
    }

    public MenuItem getMenuBack() {
        return menuBack;
    }

    public MenuBar getMenuBar() {
        return menuBar;
    }
}
```

II. AuthorProfileUpdate.java

```
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.Pane;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;

public class AuthorProfileUpdate extends Pane {
    private TextField name = new TextField();
    private TextField username = new TextField();
    private TextField password = new TextField();
    private Button update = new Button("Update Profile");

    public AuthorProfileUpdate() {
        createAuthorProfileUpdate();
    }

    public void createAuthorProfileUpdate() {
        Label edit = new Label("Please edit any or all
fields!");
        edit.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));

edit.layoutXProperty().bind(this.widthProperty().subtract(edit
.widthProperty()).divide(2));
        edit.setLayoutY(100);

        this.setStyle("-fx-background-color: antiquewhite;");

name.layoutXProperty().bind(this.widthProperty().subtract(name
.widthProperty()).divide(2));
        name.setLayoutY(200.0);

        Label nameLabel = new Label("Preferred Name");

nameLabel.layoutXProperty().bind(this.widthProperty().subtract
(name.widthProperty()).divide(2));
        nameLabel.setLayoutY(name.getLayoutY() - 20.0);



username.layoutXProperty().bind(this.widthProperty().subtract(
username.widthProperty()).divide(2));
        username.setLayoutY(name.getLayoutY() + 75.0);

        Label usernameLabel = new Label("Username");

usernameLabel.layoutXProperty().bind(this.widthProperty().subt
ract(username.widthProperty()).divide(2));
        usernameLabel.setLayoutY(username.getLayoutY() -
20.0);



password.layoutXProperty().bind(this.widthProperty().subtract(
password.widthProperty()).divide(2));
        password.setLayoutY(username.getLayoutY() + 75.0);

        Label passwordLabel = new Label("Password");

passwordLabel.layoutXProperty().bind(this.widthProperty().subt
ract(password.widthProperty()).divide(2));
        passwordLabel.setLayoutY(password.getLayoutY() -
20.0);


update.layoutXProperty().bind(this.widthProperty().subtract(pa
ssword.widthProperty()).divide(2));
        update.setLayoutY(password.getLayoutY() + 75.0);

        this.getChildren().addAll(name, nameLabel, username,
usernameLabel, password, passwordLabel, edit, update);
    }

    public TextField getUsername() {
        return username;
    }

    public TextField getName() {
        return name;
    }
    public TextField getPassword() {
        return password;
    }

    public Button getUpdate() {
        return update;
    }
}
```

III. AuthorReviewPane.java

```
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;

public class AuthorReviewPane extends BorderPane {
    private ComboBox book = new ComboBox();
    private Pane chooseBook = new Pane();
    private Manager mgr = new Manager();
    private String[] args;
    private String username;
    private Button seeReviews = new Button("See Reviews");

    public AuthorReviewPane(String username) {
        this.username = username;
        createAuthorReviewPane();
    }
    public void createAuthorReviewPane() {
        mgr.main(args);
        this.setStyle("-fx-background-color: antiquewhite;");
        Label choose = new Label("Choose book to see reviews: 
");
        choose.setFont(Font.font("Times New Roman", 
FontWeight.BOLD, 14));

choose.layoutXProperty().bind(this.widthProperty().subtract(ch
oose.widthProperty()).divide(3.25));
        choose.setLayoutY(25);
        mgr.printAuthorsBooks(username);
        for(int i = 0; i < mgr.getAuthorsBooks().size(); i++){
            String[] books = mgr.getAuthorsBooks().get(i);
            String title = books[0];
            String ISBN = books[3];
            String newBook = title + " : " + ISBN;
            book.getItems().add(newBook);
        }

book.layoutXProperty().bind(this.widthProperty().subtract(choo
se.widthProperty()).divide(2.25));
```

```
        book.setLayoutY(choose.getLayoutY() - 3);

seeReviews.layoutXProperty().bind(this.widthProperty().subtrac
t(book.widthProperty()).divide(1.45));
        seeReviews.setLayoutY(choose.getLayoutY() - 3);
        chooseBook.getChildren().addAll(book, choose,
seeReviews);
        this.setTop(chooseBook);
    }

    public ComboBox getBook() {
        return book;
    }
    public Button getSeeReviews() {
        return seeReviews;
    }
}
```

IV. BookCreationPane.java

```
import javafx.scene.control.*;
import javafx.scene.layout.Pane;

public class BookCreationPane extends Pane {

    private TextField title, author, series, volume, ISBN,
publisher, categories;
    private ComboBox genre;
    private TextArea synopsis;
    private Button createBook;
    public BookCreationPane() {
        createBookCreationPane();
    }
    public void createBookCreationPane() {
        this.setStyle("-fx-background-color: antiquewhite;");
        Label titleLabel = new Label("Title");
        titleLabel.setLayoutY(50);
        titleLabel.setLayoutX(100);
        title = new TextField();
        title.setLayoutY(titleLabel.getLayoutY()-3);
        title.setLayoutX(titleLabel.getLayoutX() + 75);
        this.getChildren().addAll(title, titleLabel);
        Label authorLabel = new Label("Author");
        authorLabel.setLayoutX(titleLabel.getLayoutX());
        authorLabel.setLayoutY((titleLabel.getLayoutY() + 50
));
        author = new TextField();
        author.setLayoutX(authorLabel.getLayoutX() + 75);
        author.setLayoutY(authorLabel.getLayoutY() - 3);
        this.getChildren().addAll(author, authorLabel);
        Label seriesLabel = new Label("Series");
        seriesLabel.setLayoutX(authorLabel.getLayoutX());
        seriesLabel.setLayoutY(authorLabel.getLayoutY() + 50);
        series = new TextField();
        series.setLayoutX(seriesLabel.getLayoutX() + 75);
        series.setLayoutY((seriesLabel.getLayoutY() - 3));
        this.getChildren().addAll(series, seriesLabel);
        Label volumeLabel = new Label("Volume");
        volumeLabel.setLayoutX(seriesLabel.getLayoutX() +
450);
        volumeLabel.setLayoutY(seriesLabel.getLayoutY());
        volume = new TextField();
        volume.setLayoutX(volumeLabel.getLayoutX() + 75);
        volume.setLayoutY(volumeLabel.getLayoutY() - 3);
        this.getChildren().addAll(volume, volumeLabel);
        Label ISBNLabel = new Label("ISBN");
        ISBNLabel.setLayoutX(seriesLabel.getLayoutX());
        ISBNLabel.setLayoutY(seriesLabel.getLayoutY() + 50);
        ISBN = new TextField();
        ISBN.setLayoutX(ISBNLabel.getLayoutX() + 75);
        ISBN.setLayoutY(ISBNLabel.getLayoutY() - 3);
        this.getChildren().addAll(ISBN, ISBNLabel);
        Label genreLabel = new Label("Genre");
        genreLabel.setLayoutX(ISBNLabel.getLayoutX());
        genreLabel.setLayoutY(ISBNLabel.getLayoutY() + 50);
        genre = new ComboBox();
        genre.getItems().addAll(
                "Romance",
                "Fantasy",
                "Mystery",
                "Thriller",
                "Historical Fiction",
                "Science Fiction"
        );
        genre.setLayoutX(genreLabel.getLayoutX() + 75);
        genre.setLayoutY(genreLabel.getLayoutY() - 3);
        this.getChildren().addAll(genre, genreLabel);
        Label categoriesLabel = new Label("Categories");
        categoriesLabel.setLayoutX(volumeLabel.getLayoutX());
        categoriesLabel.setLayoutY(volumeLabel.getLayoutY() +
100);
        categories = new TextField();
        categories.setLayoutX(categoriesLabel.getLayoutX() +
75);
        categories.setLayoutY(categoriesLabel.getLayoutY() -
3);
        this.getChildren().addAll(categories,
categoriesLabel);
        Label publisherLabel = new Label("Publisher");
        publisherLabel.setLayoutX(genreLabel.getLayoutX());
        publisherLabel.setLayoutY(genreLabel.getLayoutY() +
50);
        publisher = new TextField();
        publisher.setLayoutX(publisherLabel.getLayoutX() +
75);
        publisher.setLayoutY(publisherLabel.getLayoutY() - 3);
        this.getChildren().addAll(publisher, publisherLabel);
        Label synopsisLabel = new Label("Synopsis");
        synopsisLabel.setLayoutX(publisherLabel.getLayoutX());
        synopsisLabel.setLayoutY(publisherLabel.getLayoutY() +
50);
        synopsis = new TextArea();
        synopsis.setLayoutX(synopsisLabel.getLayoutX() + 75);
        synopsis.setLayoutY(synopsisLabel.getLayoutY() - 3);
        synopsis.setPrefWidth(600);
        synopsis.setPrefHeight(150);
        synopsis.setWrapText(true);
        this.getChildren().addAll(synopsis, synopsisLabel);
        createBook = new Button("Create Book");
        createBook.setLayoutX(425);
        createBook.setLayoutY(600);
        this.getChildren().add(createBook);
    }
    public TextField getTitle() {
        return title;
    }

    public TextField getAuthor(){
        return author;
    }

    public TextField getSeries() {
        return series;
    }

    public TextField getVolume() {
        return volume;
    }

    public TextField getISBN() {
        return ISBN;
    }

    public TextField getCategories() {
        return categories;
    }

    public ComboBox getGenre() {
        return genre;
    }

    public TextField getPublisher() {
        return publisher;
    }

    public TextArea getSynopsis() {
        return synopsis;
    }

    public Button getCreateBook() {
        return createBook;
    }
}
```

V. BookSearchPane.java

```
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;


public class BookSearchPane extends BorderPane {
    private Pane searchBar = new Pane();
    private TextField search = new TextField();
    private Button searchButton = new Button("Search");

    public BookSearchPane() {
        createBookSearchPane();
    }

    public void createBookSearchPane() {
        this.setStyle("-fx-background-color: antiquewhite;");
        Label searchBooks = new Label("Search for a book:");
        searchBooks.setFont(Font.font("Times New Roman", 
FontWeight.BOLD, 14));

searchBooks.layoutXProperty().bind(this.widthProperty().subtra
ct(searchBooks.widthProperty()).divide(4.25));
        searchBooks.setLayoutY(25);
        search = new TextField();
        search.setPrefWidth(500);

search.layoutXProperty().bind(this.widthProperty().subtract(se
archBooks.widthProperty()).divide(3));
        search.setLayoutY(searchBooks.getLayoutY() - 3);

searchButton.layoutXProperty().bind(this.widthProperty().subtr
act(searchButton.widthProperty()).divide(1.45));
        searchButton.setLayoutY(searchBooks.getLayoutY() - 3);

        searchBar.getChildren().addAll(searchBooks, search, 
searchButton);
        this.setTop(searchBar);
    }

    public TextField getSearch() {
        return search;
    }

    public Button getSearchButton() {
        return searchButton;
    }
}
```

VI. BookUpdatePane.java

```
import javafx.scene.control.*;
import javafx.scene.layout.Pane;

public class BookUpdatePane extends Pane {

    private TextField title, author, series, volume,
publisher, ISBN, categories;
    private TextArea synopsis;
    private ComboBox genre;
    private Button updateBook;
    private String isbn;
    private Manager mgr = new Manager();


    public BookUpdatePane() {
        createBookUpdatePane();
    }

    public void createBookUpdatePane() {
        this.setStyle("-fx-background-color: antiquewhite;");
        Label titleLabel = new Label("Title");
        titleLabel.setLayoutY(50);
        titleLabel.setLayoutX(100);
        title = new TextField();
        title.setLayoutY(titleLabel.getLayoutY()-3);
        title.setLayoutX(titleLabel.getLayoutX() + 75);
        this.getChildren().addAll(title, titleLabel);
        Label authorLabel = new Label("Author");
        authorLabel.setLayoutX(titleLabel.getLayoutX());
        authorLabel.setLayoutY((titleLabel.getLayoutY() + 50
));
        author = new TextField();
        author.setLayoutX(authorLabel.getLayoutX() + 75);
        author.setLayoutY(authorLabel.getLayoutY() - 3);
        this.getChildren().addAll(author, authorLabel);
        Label seriesLabel = new Label("Series");
        seriesLabel.setLayoutX(authorLabel.getLayoutX());
        seriesLabel.setLayoutY(authorLabel.getLayoutY() + 50);
        series = new TextField();
        series.setLayoutX(seriesLabel.getLayoutX() + 75);
        series.setLayoutY((seriesLabel.getLayoutY() - 3));
        series.setText(mgr.getBookDetails()[2]);
        this.getChildren().addAll(series, seriesLabel);
        Label volumeLabel = new Label("Volume");
        volumeLabel.setLayoutX(seriesLabel.getLayoutX() +
450);
        volumeLabel.setLayoutY(seriesLabel.getLayoutY());
        volume = new TextField();
        volume.setLayoutX(volumeLabel.getLayoutX() + 75);
        volume.setLayoutY(volumeLabel.getLayoutY() - 3);
        this.getChildren().addAll(volume, volumeLabel);
        Label ISBNLabel = new Label("ISBN");
        ISBNLabel.setLayoutX(seriesLabel.getLayoutX());
        ISBNLabel.setLayoutY(seriesLabel.getLayoutY() + 50);
        ISBN = new TextField();
        ISBN.setLayoutX(ISBNLabel.getLayoutX() + 75);
        ISBN.setLayoutY(ISBNLabel.getLayoutY() - 3);
        this.getChildren().addAll(ISBN, ISBNLabel);
        Label genreLabel = new Label("Genre");
        genreLabel.setLayoutX(ISBNLabel.getLayoutX());
        genreLabel.setLayoutY(ISBNLabel.getLayoutY() + 50);
        genre = new ComboBox();
        genre.getItems().addAll(
                "Romance",
                "Fantasy",
                "Mystery",
                "Thriller",
                "Historical Fiction",
                "Science Fiction"
        );
        genre.setLayoutX(genreLabel.getLayoutX() + 75);
        genre.setLayoutY(genreLabel.getLayoutY() - 3);
        this.getChildren().addAll(genre, genreLabel);
        Label categoriesLabel = new Label("Categories");
        categoriesLabel.setLayoutX(volumeLabel.getLayoutX());
        categoriesLabel.setLayoutY(volumeLabel.getLayoutY() +
100);
        categories = new TextField();
        categories.setLayoutX(categoriesLabel.getLayoutX() +
75);
        categories.setLayoutY(categoriesLabel.getLayoutY() -
3);
        this.getChildren().addAll(categories,
categoriesLabel);
        Label publisherLabel = new Label("Publisher");
        publisherLabel.setLayoutX(genreLabel.getLayoutX());
        publisherLabel.setLayoutY(genreLabel.getLayoutY() +
50);
        publisher = new TextField();
        publisher.setLayoutX(publisherLabel.getLayoutX() +
75);
        publisher.setLayoutY(publisherLabel.getLayoutY() - 3);
        this.getChildren().addAll(publisher, publisherLabel);
        Label synopsisLabel = new Label("Synopsis");
        synopsisLabel.setLayoutX(publisherLabel.getLayoutX());
        synopsisLabel.setLayoutY(publisherLabel.getLayoutY() +
50);
        synopsis = new TextArea();
        synopsis.setLayoutX(synopsisLabel.getLayoutX() + 75);
        synopsis.setLayoutY(synopsisLabel.getLayoutY() - 3);
        synopsis.setPrefWidth(600);
        synopsis.setPrefHeight(150);
        synopsis.setWrapText(true);
        this.getChildren().addAll(synopsis, synopsisLabel);
        updateBook = new Button("Update Book");
        updateBook.setLayoutX(425);
        updateBook.setLayoutY(600);
        this.getChildren().add(updateBook);
    }

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

    }

    public TextField getTitle() {
        return title;
    }

    public TextField getAuthor() {
        return author;
    }

    public TextField getSeries() {
        return series;
    }

    public TextField getVolume() {
        return volume;
    }

    public TextField getPublisher() {
        return publisher;
    }

    public TextField getISBN() {
        return ISBN;
    }

    public TextField getCategories() {
        return categories;
    }

    public TextArea getSynopsis() {
        return synopsis;
    }

    public Button getUpdateBook() {
        return updateBook;
    }
    public String getIsbn() {
        return isbn;
    }

    public ComboBox getGenre() {
        return genre;
    }
}
```

VII. LoginScreen.java

```
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.Pane;

public class LoginScreen extends Pane {
    private Button login = new Button("Login");
    private Button newUser = new Button("Click here to create
a new user");
    private TextField username = new TextField("Username");
    private TextField password = new TextField("Password");

    public LoginScreen() {
        createHomeScreen();
    }
    public void createHomeScreen() {
        this.setStyle("-fx-background-color: antiquewhite;");
        Image image = new
Image(getClass().getResourceAsStream("BookNookLogo.png"));
        ImageView view = new ImageView();
        view.setImage(image);

view.layoutXProperty().bind(this.widthProperty().subtract(view
.getFitWidth()).divide(3));
        view.setLayoutY(100.0);
        view.setFitHeight(500);
        view.setFitWidth(500);
        view.setSmooth(true);
        view.setCache(true);
        view.setPreserveRatio(true);


username.layoutXProperty().bind(this.widthProperty().subtract(
username.widthProperty()).divide(2));
        username.setLayoutY(view.getLayoutY() + 150.0);


password.layoutXProperty().bind(this.widthProperty().subtract(
password.widthProperty()).divide(2));
        password.setLayoutY(username.getLayoutY() + 35.0);


newUser.layoutXProperty().bind(this.widthProperty().subtract(p
assword.widthProperty()).divide(2.25));
        newUser.setLayoutY(password.getLayoutY() + 75.0);


login.layoutXProperty().bind(this.widthProperty().subtract(pas
sword.widthProperty()).divide(1.65));
        login.setLayoutY(newUser.getLayoutY());

        this.getChildren().add(view);
        this.getChildren().addAll(username, password);
        this.getChildren().addAll(newUser, login);
    }

    public Button getNewUser() {
        return newUser;
    }
    public Button getLogin() {
        return login;
    }
    public TextField getUsername() {
        return username;
    }
    public TextField getPassword() {
        return password;
    }
}
```

VIII. MainUI.java

```
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Pos;
import javafx.geometry.Rectangle2D;
import javafx.geometry.Side;
import javafx.scene.Scene;
import javafx.scene.chart.PieChart;
import javafx.scene.control.*;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.stage.Popup;
import javafx.stage.Screen;
import javafx.stage.Stage;
import java.util.ArrayList;

public class MainUI extends Application {
    private BorderPane pane = new BorderPane();
    private GridPane authorPane = new GridPane();

    private GridPane readerPane = new GridPane();
    private LoginScreen loginPane = new LoginScreen();
    private String username;



    private AuthorHomeScreen authorHomeScreen = new
AuthorHomeScreen();
    private AuthorProfileUpdate authorProfileUpdate = new
AuthorProfileUpdate();
    private ReaderHomeScreen readerHomeScreen = new
ReaderHomeScreen();
    private ReaderProfileUpdate readerProfileUpdate = new
ReaderProfileUpdate();
    private Manager mgr = new Manager();
    private String[] args;


    private RegistrationPane registrationPane = new
RegistrationPane();

    private BookCreationPane bookCreationPane = new
BookCreationPane();
    private BookUpdatePane bookUpdatePane = new
BookUpdatePane();
    private BookSearchPane bookSearchPane = new
BookSearchPane();
    private ReaderReviewPane readerReviewPane = new
```

```
ReaderReviewPane();
    private Popup updatePopup = new Popup();
    private Popup deletePopup = new Popup();
    private Popup deleteProfilePopup = new Popup();
    private ReadingStatusPane readingStatusPane = new
ReadingStatusPane();
    private String readingStatusISBN;
    final PieChart storylineChart = new PieChart();
    final PieChart plotChart = new PieChart();
    final PieChart settingChart = new PieChart();
    final PieChart spiceChart = new PieChart();
    final PieChart charactersChart = new PieChart();
    final PieChart wbChart = new PieChart();
    final PieChart wsChart = new PieChart();
    final PieChart genres = new PieChart();


    int user;



    public MainUI() {
        createGUI();
        mgr.main(args);
    }

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage stage) {
        stage.setTitle("Book Nook");
        Scene scene = new Scene(pane, 1000, 750);
        stage.setScene(scene);
        Rectangle2D screen =
Screen.getPrimary().getVisualBounds();
        stage.setX(screen.getMinX());
        stage.setY(screen.getMinY());
        stage.setWidth(screen.getWidth());
        stage.setHeight(screen.getHeight());
        stage.show();

        authorHomeScreen.getMenuUpdate().setOnAction(e4 -> {
            updatePopup.getContent().clear();
            deletePopup.hide();
            createUpdate();
            updatePopup.show(stage);
            authorHomeScreen.setCenter(null);

        });

        authorHomeScreen.getMenuDelete().setOnAction(e5 -> {
            deletePopup.getContent().clear();
            updatePopup.hide();
            deletePopup.setHeight(200);
            deletePopup.setWidth(200);
            Label text = new Label("Please enter the ISBN of
the book to delete:");
            TextField ISBN = new TextField();
            Button delete = new Button("Delete");
            delete.setLayoutX(ISBN.getLayoutX() + 148);
            delete.setLayoutY(ISBN.getLayoutY() + 100);
            text.setLayoutX(65);
            text.setLayoutY(0);
            ISBN.setLayoutX(100);
            ISBN.setLayoutY(text.getLayoutY() + 50);
            deletePopup.getContent().addAll(text, ISBN,
delete);
            deletePopup.setAnchorX(575);
            deletePopup.setAnchorY(325);
            deletePopup.show(stage);
            authorHomeScreen.setCenter(null);
            delete.setOnAction(e -> {
                if (ISBN.getText().length() != 13) {
                    Alert error = new
Alert(Alert.AlertType.ERROR);
                    error.setTitle("Uh oh...");
                    error.setHeaderText("Invalid ISBN");
                    error.setContentText("Please enter a valid
ISBN. They are strings of 13 numbers.");
                    error.show();
                    error.setY(500);
                } else {
                    String isbn = ISBN.getText();
                    deletePopup.hide();
                    mgr.deleteBook(isbn);
                    removeBooks();
                    printBooks();
                }
            });

        });

        authorHomeScreen.getMenuDeleteProfile().setOnAction(e9
-> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.getContent().clear();
            createDeletePopup();
            deleteProfilePopup.show(stage);
        });

        readerHomeScreen.getMenuUpdateReview().setOnAction(e19
-> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();
            updatePopup.getContent().clear();
            createUpdate();
            updatePopup.show(stage);
            readerHomeScreen.setCenter(null);
        });

        readerHomeScreen.getMenuDeleteReview().setOnAction(e -
>  {
            deletePopup.getContent().clear();
            updatePopup.hide();
            deletePopup.setHeight(200);
            deletePopup.setWidth(200);
            Label text = new Label("Please enter the ISBN of
the review to delete:");
            TextField ISBN = new TextField();
            Button delete = new Button("Delete");
            delete.setLayoutX(ISBN.getLayoutX() + 148);
            delete.setLayoutY(ISBN.getLayoutY() + 100);
            text.setLayoutX(65);
            text.setLayoutY(0);
            ISBN.setLayoutX(100);
            ISBN.setLayoutY(text.getLayoutY() + 50);
            deletePopup.getContent().addAll(text, ISBN,
delete);
            deletePopup.setAnchorX(575);
            deletePopup.setAnchorY(325);
            deletePopup.show(stage);
            readerHomeScreen.setCenter(null);
            delete.setOnAction(e2 -> {
                if (ISBN.getText().length() != 13) {
                    Alert error = new
Alert(Alert.AlertType.ERROR);
                    error.setTitle("Uh oh...");
                    error.setHeaderText("Invalid ISBN");
                    error.setContentText("Please enter a valid
ISBN. They are strings of 13 numbers.");
                    error.show();
                    error.setY(500);
                } else {
                    deletePopup.hide();
                    mgr.deleteReview(username,
ISBN.getText());
                    removeBooks();
                    printReadBooks();
                    readerHomeScreen.setCenter(readerPane);
                    readerPane.requestFocus();
                }
            });
        });

        readerHomeScreen.getMenuDeleteProfile().setOnAction(e9
-> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.getContent().clear();
            createDeletePopup();
            deleteProfilePopup.show(stage);
        });

        authorHomeScreen.getMenuReviews().setOnAction((e7 -> {
            AuthorReviewPane authorReview = new
AuthorReviewPane(username);
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();
            authorHomeScreen.setCenter(authorReview);
            authorReview.requestFocus();

            authorReview.getSeeReviews().setOnAction(e8 -> {
                String[] book =
authorReview.getBook().getValue().toString().split(" : ");
                ArrayList<int[]> reviews =
mgr.getAuthorReviews(book[1]);
                Pane charts = new Pane();
                createCharts(reviews);
                charts.getChildren().addAll(storylineChart,
plotChart, settingChart, spiceChart, charactersChart, wbChart,
wsChart);
                authorReview.setCenter(charts);
                charts.requestFocus();
                final Label caption = new Label("");
                caption.setTextFill(Color.BLACK);
                caption.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                Label storylineText = new Label("Storyline");
                storylineText.setFont(Font.font("Times New
Roman", FontWeight.BOLD, 14));
                storylineText.setLayoutX(82);
                storylineText.setLayoutY(10);
                charts.getChildren().add(storylineText);
                for (final PieChart.Data data :
storylineChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e -> {

caption.setLayoutX(storylineChart.getLayoutX() + 210);

caption.setLayoutY(storylineChart.getLayoutY() + 93);

caption.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption);
                            });
```

```
data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption);
                    });
                }

                Label plotText = new Label("Plot");
                plotText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                plotText.setLayoutX(storylineText.getLayoutX()
+ 15);
                plotText.setLayoutY(storylineText.getLayoutY()
+ 210);
                charts.getChildren().add(plotText);
                final Label caption2 = new Label("");
                caption2.setTextFill(Color.BLACK);
                caption2.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
plotChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e -> {

caption2.setLayoutX(plotChart.getLayoutX() + 210);

caption2.setLayoutY(plotChart.getLayoutY() + 93);

caption2.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption2);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption2);
                    });
                }

                Label spiceText = new Label("Spice");
                spiceText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));

spiceText.setLayoutX(storylineText.getLayoutX() + 610);

spiceText.setLayoutY(storylineText.getLayoutY());
                charts.getChildren().add(spiceText);
                final Label caption4 = new Label("");
                caption4.setTextFill(Color.BLACK);
                caption4.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
spiceChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e -> {

caption4.setLayoutX(spiceChart.getLayoutX() + 210);

caption4.setLayoutY(spiceChart.getLayoutY() + 93);

caption4.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption4);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption4);
                    });
                }

                Label charactersText = new
Label("Characters");
                charactersText.setFont(Font.font("Times New
Roman", FontWeight.BOLD, 14));

charactersText.setLayoutX(plotText.getLayoutX() + 575);

charactersText.setLayoutY(plotText.getLayoutY());
                charts.getChildren().add(charactersText);
                final Label caption5 = new Label("");
                caption5.setTextFill(Color.BLACK);
                caption5.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
charactersChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e -> {

caption5.setLayoutX(charactersChart.getLayoutX() + 210);

caption5.setLayoutY(charactersChart.getLayoutY() + 93);

caption5.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption5);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption5);
                    });
                }

                Label settingText = new Label("Setting");
                settingText.setFont(Font.font("Times New
Roman", FontWeight.BOLD, 14));

settingText.setLayoutX(charactersText.getLayoutX() + 17);

settingText.setLayoutY(charactersText.getLayoutY() + 210);
                charts.getChildren().add(settingText);
                final Label caption3 = new Label("");
                caption3.setTextFill(Color.BLACK);
                caption3.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
settingChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e -> {

caption3.setLayoutX(settingChart.getLayoutX() + 210);

caption3.setLayoutY(settingChart.getLayoutY() + 93);

caption3.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption3);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption3);
                    });
                }

                Label wbText = new Label("World Building");
                wbText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                wbText.setLayoutX(spiceText.getLayoutX() +
565);
                wbText.setLayoutY(spiceText.getLayoutY());
                charts.getChildren().add(wbText);
                final Label caption6 = new Label("");
                caption6.setTextFill(Color.BLACK);
                caption6.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
wbChart.getData()) {
```

```
data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e -> {

caption6.setLayoutX(wbChart.getLayoutX() + 210);

caption6.setLayoutY(wbChart.getLayoutY() + 93);

caption6.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption6);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption6);
                    });
                }

                Label wsText = new Label("Writing Style");
                wsText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                wsText.setLayoutX(charactersText.getLayoutX()
+ 595);

wsText.setLayoutY(charactersText.getLayoutY());
                charts.getChildren().add(wsText);
                final Label caption7 = new Label("");
                caption7.setTextFill(Color.BLACK);
                caption7.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
wsChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e -> {

caption7.setLayoutX(wsChart.getLayoutX() + 210);

caption7.setLayoutY(wsChart.getLayoutY() + 93);

caption7.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption7);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption7);
                    });
                }
```

```
            });
        }));

        readerHomeScreen.getMenuSeeReviews().setOnAction(e ->
{
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();

readerReviewPane.setTop(readerReviewPane.getPane());
            readerReviewPane.setCenter(null);
            readerHomeScreen.setCenter(readerReviewPane);
            readerReviewPane.requestFocus();

            readerReviewPane.getGetReviews().setOnAction(e8 ->
{
                String dsMonth =
readerReviewPane.getDsMonth().getText();
                String dsDay =
readerReviewPane.getDsDay().getText();
                String dsYear =
readerReviewPane.getDsYear().getText();
                String dateStart = dsYear + "-" + dsMonth + "-
" + dsDay;
                String dfMonth =
readerReviewPane.getDfMonth().getText();
                String dfDay =
readerReviewPane.getDfDay().getText();
                String dfYear =
readerReviewPane.getDfYear().getText();
                String dateFinish = dfYear + "-" + dfMonth +
"-" + dfDay;
                ArrayList<int[]> reviews =
mgr.getReaderReviews(username, dateStart, dateFinish);
                Pane charts = new Pane();
                createCharts(reviews);
                charts.getChildren().addAll(storylineChart,
plotChart, settingChart, spiceChart, charactersChart, wbChart,
wsChart);
                readerReviewPane.setCenter(charts);
                charts.requestFocus();
                final Label caption = new Label("");
                caption.setTextFill(Color.BLACK);
                caption.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                Label storylineText = new Label("Storyline");
                storylineText.setFont(Font.font("Times New
Roman", FontWeight.BOLD, 14));
                storylineText.setLayoutX(82);
                storylineText.setLayoutY(10);
                charts.getChildren().add(storylineText);
                for (final PieChart.Data data :
storylineChart.getData()) {
```

```
data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e1 -> {

caption.setLayoutX(storylineChart.getLayoutX() + 210);

caption.setLayoutY(storylineChart.getLayoutY() + 93);

caption.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption);
                    });
                }

                Label plotText = new Label("Plot");
                plotText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                plotText.setLayoutX(storylineText.getLayoutX()
+ 15);
                plotText.setLayoutY(storylineText.getLayoutY()
+ 210);
                charts.getChildren().add(plotText);
                final Label caption2 = new Label("");
                caption2.setTextFill(Color.BLACK);
                caption2.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
plotChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e1 -> {

caption2.setLayoutX(plotChart.getLayoutX() + 210);

caption2.setLayoutY(plotChart.getLayoutY() + 93);

caption2.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption2);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption2);
                    });
                }
                Label spiceText = new Label("Spice");
                spiceText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));

spiceText.setLayoutX(storylineText.getLayoutX() + 610);

spiceText.setLayoutY(storylineText.getLayoutY());
                charts.getChildren().add(spiceText);
                final Label caption4 = new Label("");
                caption4.setTextFill(Color.BLACK);
                caption4.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
spiceChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e1 -> {

caption4.setLayoutX(spiceChart.getLayoutX() + 210);

caption4.setLayoutY(spiceChart.getLayoutY() + 93);

caption4.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption4);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption4);
                    });
                }

                Label charactersText = new
Label("Characters");
                charactersText.setFont(Font.font("Times New
Roman", FontWeight.BOLD, 14));

charactersText.setLayoutX(plotText.getLayoutX() + 575);

charactersText.setLayoutY(plotText.getLayoutY());
                charts.getChildren().add(charactersText);
                final Label caption5 = new Label("");
                caption5.setTextFill(Color.BLACK);
                caption5.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
charactersChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e1 -> {
```

```
caption5.setLayoutX(charactersChart.getLayoutX() + 210);

caption5.setLayoutY(charactersChart.getLayoutY() + 93);

caption5.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption5);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption5);
                    });
                }

                Label settingText = new Label("Setting");
                settingText.setFont(Font.font("Times New
Roman", FontWeight.BOLD, 14));

settingText.setLayoutX(charactersText.getLayoutX() + 17);

settingText.setLayoutY(charactersText.getLayoutY() + 210);
                charts.getChildren().add(settingText);
                final Label caption3 = new Label("");
                caption3.setTextFill(Color.BLACK);
                caption3.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
settingChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e1 -> {

caption3.setLayoutX(settingChart.getLayoutX() + 210);

caption3.setLayoutY(settingChart.getLayoutY() + 93);

caption3.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption3);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption3);
                    });
                }

                Label wbText = new Label("World Building");
                wbText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                wbText.setLayoutX(spiceText.getLayoutX() +
565);
                wbText.setLayoutY(spiceText.getLayoutY());
                charts.getChildren().add(wbText);
                final Label caption6 = new Label("");
                caption6.setTextFill(Color.BLACK);
                caption6.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
wbChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e1 -> {

caption6.setLayoutX(wbChart.getLayoutX() + 210);

caption6.setLayoutY(wbChart.getLayoutY() + 93);

caption6.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption6);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption6);
                    });
                }

                Label wsText = new Label("Writing Style");
                wsText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                wsText.setLayoutX(charactersText.getLayoutX()
+ 595);

wsText.setLayoutY(charactersText.getLayoutY());
                charts.getChildren().add(wsText);
                final Label caption7 = new Label("");
                caption7.setTextFill(Color.BLACK);
                caption7.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
wsChart.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e1 -> {

caption7.setLayoutX(wsChart.getLayoutX() + 210);

caption7.setLayoutY(wsChart.getLayoutY() + 93);
caption7.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption7);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption7);
                    });
                }
                Label num = new Label("You've read " +
mgr.getNumberBooks(username, dateStart, dateFinish) + "
book(s) between " + dsMonth + "/" + dsDay + "/" + dsYear + "
and " + dfMonth + "/" + dfDay + "/" + dfYear + "!");
                num.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 24));
                num.setLayoutY(10);
                Pane top = new Pane();

num.layoutXProperty().bind(top.widthProperty().subtract(num.wi
dthProperty()).divide(2));
                top.getChildren().add(num);
                readerReviewPane.setTop(top);
                readerReviewPane.getDsDay().setText("");
                readerReviewPane.getDsMonth().setText("");
                readerReviewPane.getDsYear().setText("");
                readerReviewPane.getDfDay().setText("");
                readerReviewPane.getDfMonth().setText("");
                readerReviewPane.getDfYear().setText("");

                int[] g = reviews.get(7);
                ObservableList<PieChart.Data> genreData =
                        FXCollections.observableArrayList(
                                new PieChart.Data("Romance",
g[0]),
                                new PieChart.Data("Fantasy",
g[1]),
                                new PieChart.Data("Mystery",
g[2]),
                                new PieChart.Data("Thriller",
g[3]),
                                new PieChart.Data("Historical
Fiction", g[4]),
                                new PieChart.Data("Science
Fiction", g[5]));
                genres.setMaxSize(200, 200);
                genres.setData(genreData);
                genres.setClockwise(true);
                genres.setLabelsVisible(true);
                genres.setLabelLineLength(200);
                genres.setLegendVisible(false);
                genres.setLegendSide(Side.RIGHT);
                genres.setLayoutX(plotChart.getLayoutX() +
275);
                genres.setLayoutY(plotChart.getLayoutY() +
210);

                charts.getChildren().add(genres);

settingChart.setLayoutX(charactersChart.getLayoutX() + 275);

settingChart.setLayoutY(charactersChart.getLayoutY() + 210);

settingText.setLayoutY(charactersText.getLayoutY() + 210);

settingText.setLayoutX(charactersText.getLayoutX() + 293);

                Label genreText = new Label("Genres");
                genreText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                genreText.setLayoutX(plotText.getLayoutX() +
265);

genreText.setLayoutY(charactersText.getLayoutY() + 210);
                charts.getChildren().add(genreText);
                final Label caption8 = new Label("");
                caption8.setTextFill(Color.BLACK);
                caption8.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
                for (final PieChart.Data data :
genres.getData()) {

data.getNode().addEventHandler(MouseEvent.MOUSE_ENTERED,
                            e1 -> {

caption8.setLayoutX(genres.getLayoutX() + 210);

caption8.setLayoutY(genres.getLayoutY() + 93);

caption8.setText(String.valueOf((int) data.getPieValue()));

charts.getChildren().add(caption8);
                            });


data.getNode().addEventHandler(MouseEvent.MOUSE_EXITED, e1 ->
{
                        charts.getChildren().remove(caption8);
                    });
                }

            });
        });

    }
    public void createGUI() {
        pane.setStyle("-fx-background-color: antiquewhite;");
        pane.setCenter(loginPane);
        loginPane.requestFocus();
        loginPane.getNewUser().setOnAction(e -> {
            pane.setCenter(registrationPane);
            registrationPane.requestFocus();

        });
        registrationPane.getRegister().setOnAction(e -> {
            if(registrationPane.getAuthor().isSelected()){
                user = 1;
            } else
if(registrationPane.getReader().isSelected()){
                user = 2;
            }
            int duplicate = mgr.verify(user,
registrationPane.getUsername().getText());
            if (duplicate == 0) {

mgr.createUser(registrationPane.getUsername().getText(),
registrationPane.getPassword().getText(),
registrationPane.getName().getText(), user);
                pane.setCenter(loginPane);
                loginPane.requestFocus();
                Alert welcome = new
Alert(Alert.AlertType.INFORMATION);
                welcome.setHeaderText("Welcome to Book
Nook!");
                welcome.setContentText("Your user was created!
Please login to your profile.");
                welcome.show();
            } else {
                Alert taken = new
Alert(Alert.AlertType.ERROR);
                taken.setTitle("Uh Oh...");
                taken.setHeaderText("User Taken");
                taken.setContentText("The username you entered
is already taken. Please change the username and try again.");
                taken.show();
                registrationPane.getUsername().setText("");
                registrationPane.getPassword().setText("");
                registrationPane.getName().setText("");

registrationPane.getAuthor().setSelected(false);

registrationPane.getReader().setSelected(false);
            }

        });

        loginPane.getLogin().setOnAction(e -> {
            user =
mgr.checkUser(loginPane.getUsername().getText(),
loginPane.getPassword().getText());
            if (user == 1) {

authorHomeScreen.setTop(authorHomeScreen.getMenuBar());
                username = loginPane.getUsername().getText();
                pane.setCenter(authorHomeScreen);
                authorHomeScreen.requestFocus();
                printBooks();
            } else if (user == 2) {

readerHomeScreen.setTop(readerHomeScreen.getMenu());
                username = loginPane.getUsername().getText();
                pane.setCenter(readerHomeScreen);
                readerHomeScreen.requestFocus();
                printReadBooks();
            } else {
                Alert error = new
Alert(Alert.AlertType.ERROR);
                error.setTitle("Wrong username or password!");
                error.setHeaderText("Wrong username or
password!");
                error.setContentText(("Please try again."));
                error.show();
            }

        });

        authorHomeScreen.getMenuLogout().setOnAction(e1 -> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();
            loginPane.getUsername().setText("Username");
            loginPane.getPassword().setText("Password");
            removeBooks();
            authorHomeScreen.setCenter(loginPane);
            authorHomeScreen.setTop(null);
            loginPane.requestFocus();
        });

        authorHomeScreen.getMenuCreate().setOnAction(e2 -> {
            deleteProfilePopup.hide();
            updatePopup.hide();
            deletePopup.hide();
            authorHomeScreen.setCenter(bookCreationPane);
            bookCreationPane.requestFocus();
        });

        bookCreationPane.getCreateBook().setOnAction(e3 -> {
            if (bookCreationPane.getISBN().getText().length()
!= 13) {
                Alert error = new
```

```
Alert(Alert.AlertType.ERROR);
                error.setTitle("Uh oh...");
                error.setHeaderText("Invalid ISBN");
                error.setContentText("Please enter a valid
ISBN. They are strings of 13 numbers.");
                error.show();
            } else {
                String title =
bookCreationPane.getTitle().getText();
                String author =
bookCreationPane.getAuthor().getText();
                String series =
bookCreationPane.getSeries().getText();
                Float volume =
Float.parseFloat(bookCreationPane.getVolume().getText());
                String ISBN =
bookCreationPane.getISBN().getText();
                String genre =
String.valueOf(bookCreationPane.getGenre().getValue());
                String categories =
bookCreationPane.getCategories().getText();
                String publisher =
bookCreationPane.getPublisher().getText();
                String synopsis =
bookCreationPane.getSynopsis().getText();
                mgr.createBook(title, author, series, volume,
ISBN, genre, categories, publisher, synopsis, username);
                bookCreationPane.getTitle().setText("");
                bookCreationPane.getAuthor().setText("");
                bookCreationPane.getSeries().setText("");
                bookCreationPane.getVolume().setText("");
                bookCreationPane.getGenre().setValue("");
                bookCreationPane.getISBN().setText("");
                bookCreationPane.getCategories().setText("");
                bookCreationPane.getPublisher().setText("");
                bookCreationPane.getSynopsis().setText("");
                removeBooks();
                printBooks();
            }
        });

        bookUpdatePane.getUpdateBook().setOnAction(e5 -> {
            if (bookUpdatePane.getISBN().getText().length() !=
13) {
                Alert error = new
Alert(Alert.AlertType.ERROR);
                error.setTitle("Uh oh...");
                error.setHeaderText("Invalid ISBN");
                error.setContentText("Please enter a valid
ISBN. They are strings of 13 numbers.");
                error.show();
                error.setY(500);
            } else {
                String title =
bookUpdatePane.getTitle().getText();
                String author =
bookUpdatePane.getAuthor().getText();
                String series =
bookUpdatePane.getSeries().getText();
                Float volume =
Float.parseFloat(bookUpdatePane.getVolume().getText());
                String ISBN =
bookUpdatePane.getISBN().getText();
                String genre =
String.valueOf(bookUpdatePane.getGenre().getValue());
                String categories =
bookUpdatePane.getCategories().getText();
                String publisher =
bookUpdatePane.getPublisher().getText();
                String synopsis =
bookUpdatePane.getSynopsis().getText();
                String oldISBN = bookUpdatePane.getIsbn();
                mgr.updateBook(title, author, series, volume,
ISBN, genre, categories, publisher, synopsis, oldISBN,
username);
                removeBooks();
                printBooks();
            }
        });

        authorHomeScreen.getMenuBack().setOnAction(e6 -> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();
            removeBooks();
            printBooks();
        });


        authorHomeScreen.getMenuUpdateProfile().setOnAction(e8
-> {
            updatePopup.hide();
            deletePopup.hide();
            fillProfile();
            authorHomeScreen.setCenter(authorProfileUpdate);
            authorProfileUpdate.requestFocus();
        });

        authorProfileUpdate.getUpdate().setOnAction(e9 -> {
            String name =
authorProfileUpdate.getName().getText();
            String newUsername =
authorProfileUpdate.getUsername().getText();
            String password =
authorProfileUpdate.getPassword().getText();
            mgr.updateAuthor(name, newUsername, password,
username);
            mgr.setNameOfUser(name);
            removeBooks();
            printBooks();
            authorHomeScreen.setCenter(authorPane);
            authorPane.requestFocus();
            Alert updated = new
Alert(Alert.AlertType.CONFIRMATION);
            updated.setHeaderText("Updated!");
            updated.setContentText("Your profile information
was successfully updated!");
            updated.show();
        });

        readerHomeScreen.getMenuLogout().setOnAction(e10 -> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();
            loginPane.getUsername().setText("Username");
            loginPane.getPassword().setText("Password");
            readerHomeScreen.setCenter(loginPane);
            readerHomeScreen.setTop(null);

readerReviewPane.setTop(readerReviewPane.getPane());
            readerReviewPane.setCenter(null);
            removeBooks();
            loginPane.requestFocus();
            readerReviewPane.getDsMonth().setText("");
            readerReviewPane.getDsYear().setText("");
            readerReviewPane.getDsDay().setText("");
            readerReviewPane.getDfMonth().setText("");
            readerReviewPane.getDfYear().setText("");
            readerReviewPane.getDfDay().setText("");
        });

        readerHomeScreen.getMenuRead().setOnAction(e11 -> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();
            removeBooks();
            printReadBooks();
        });

        readerHomeScreen.getMenuReading().setOnAction(e12 -> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();
            removeBooks();
            printReadingBooks();
        });

        readerHomeScreen.getMenuTBR().setOnAction(e13 -> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();
            removeBooks();
            printTBR();
        });

        readerHomeScreen.getMenuBook().setOnAction(e14 -> {
            updatePopup.hide();
            deletePopup.hide();
            deleteProfilePopup.hide();
            readerHomeScreen.setCenter(bookSearchPane);
        });

        bookSearchPane.getSearchButton().setOnAction(e15 -> {
            searchBooks();
        });

        readingStatusPane.getSearch().setOnAction(e16 -> {
            readingStatusPane.getReadingStatus().setValue("");
            readingStatusPane.getSearch().setVisible(false);

readingStatusPane.getSetReadingStatus().setVisible(true);

readingStatusPane.getReadingStatus().setVisible(true);

readingStatusPane.getSetReadingStatusText().setVisible(true);

readingStatusPane.getEditReadingStatus().setVisible(true);

readingStatusPane.getRemoveStatus().setVisible(true);
            String status = mgr.getReadingStatus(username,
readingStatusISBN);

readingStatusPane.getReadingStatus().setValue(status);
        });

        readingStatusPane.getSetReadingStatus().setOnAction(e17
-> {
            if (readingStatusPane.getISBN().getText().length()
!= 13) {
                Alert error = new
Alert(Alert.AlertType.ERROR);
                error.setTitle("Uh oh...");
                error.setHeaderText("Invalid ISBN");
                error.setContentText("Please enter a valid
ISBN. They are strings of 13 numbers.");
                error.show();
            } else {
               mgr.setReadingStatus(username,
readingStatusPane.getISBN().getText(),
String.valueOf(readingStatusPane.getReadingStatus().getValue()
));
               removeBooks();
               printReadBooks();

if(String.valueOf(readingStatusPane.getReadingStatus().getValu
e()).equalsIgnoreCase("read")){
                   createReviewPane();
               } else {
                   readerHomeScreen.setCenter(readerPane);
                   readerPane.requestFocus();
               }

readingStatusPane.getReadingStatus().setValue("");
           }

       });


readingStatusPane.getEditReadingStatus().setOnAction(e18 -> {
           mgr.updateReadingStatus(username,
readingStatusPane.getISBN().getText(),
String.valueOf(readingStatusPane.getReadingStatus().getValue()
));
           removeBooks();
           printReadBooks();

if(String.valueOf(readingStatusPane.getReadingStatus().getValu
e()).equalsIgnoreCase("read")){
               createReviewPane();
           } else {
               readerHomeScreen.setCenter(readerPane);
               readerPane.requestFocus();
           }
           readingStatusPane.getReadingStatus().setValue("");
       });

       readingStatusPane.getRemoveStatus().setOnAction(e -> {
           mgr.removeReadingStatus(username,
readingStatusPane.getISBN().getText());
           removeBooks();
           printReadBooks();
           readerHomeScreen.setCenter(readerPane);
       });

        readerHomeScreen.getMenuUpdateProfile().setOnAction(e8
-> {
            updatePopup.hide();
            deletePopup.hide();
            fillProfile();
            readerHomeScreen.setCenter(readerProfileUpdate);
            readerProfileUpdate.requestFocus();
        });

        readerProfileUpdate.getUpdate().setOnAction(e -> {
            String name =
readerProfileUpdate.getName().getText();
            String newUsername =
readerProfileUpdate.getUsername().getText();
            String password =
readerProfileUpdate.getPassword().getText();
            mgr.updateReader(name, newUsername, password,
username);
            mgr.setNameOfUser(name);
            removeBooks();
            printReadBooks();
            readerHomeScreen.setCenter(readerPane);
            authorPane.requestFocus();
            Alert updated = new
Alert(Alert.AlertType.CONFIRMATION);
            updated.setHeaderText("Updated!");
            updated.setContentText("Your profile information
was successfully updated!");
            updated.show();
        });


    }

    public void fillProfile() {
        String[] profile;
        if(user == 1) {
            profile = mgr.printProfile(username, user);
            authorProfileUpdate.getName().setText(profile[0]);

authorProfileUpdate.getUsername().setText(profile[1]);

authorProfileUpdate.getPassword().setText(profile[2]);
        } else {
            profile = mgr.printProfile(username, user);
            readerProfileUpdate.getName().setText(profile[0]);

readerProfileUpdate.getUsername().setText(profile[1]);

readerProfileUpdate.getPassword().setText(profile[2]);
        }

    }


    public void fillBoxes(){

bookUpdatePane.getTitle().setText(mgr.getBookDetails()[0]);

bookUpdatePane.getAuthor().setText(mgr.getBookDetails()[1]);

bookUpdatePane.getSeries().setText(mgr.getBookDetails()[2]);

bookUpdatePane.getVolume().setText(String.valueOf(mgr.getBookV
olume()));

bookUpdatePane.getISBN().setText(bookUpdatePane.getIsbn());

bookUpdatePane.getGenre().setValue(mgr.getBookDetails()[3]);

bookUpdatePane.getCategories().setText(mgr.getBookDetails()[4]
);

bookUpdatePane.getPublisher().setText(mgr.getBookDetails()[5])
;

bookUpdatePane.getSynopsis().setText(mgr.getBookDetails()[6]);
    }

    public void removeBooks() {
        if(user == 1) {
            authorPane.getChildren().clear();
        } else {
            readerPane.getChildren().clear();
            mgr.getTBRBooks().clear();
            mgr.getReadBooks().clear();
            mgr.getReadingBooks().clear();
        }


    }

    public void printBooks() {
        Label name = new Label(mgr.getNameOfUser() + "'s
Books");
        name.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 24));
        authorPane.setStyle("-fx-background-color:
antiquewhite;");
        authorPane.add(name, 0, 0);
        Label title = new Label("Title");
        title.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label series = new Label("Series");
        series.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label volume = new Label("Volume");
        volume.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label isbn = new Label("ISBN");
        isbn.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label publisher = new Label("Publisher");
        publisher.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label blank = new Label("");
        authorPane.add(blank, 0, 1);
        authorPane.setHgap(125);
        authorPane.add(title, 0,2);
        authorPane.add(series, 1,2);
        authorPane.add(volume, 2,2);
        authorPane.add(isbn, 3,2);
        authorPane.add(publisher, 4,2);
        mgr.printAuthorsBooks(username);
        int j = 3;
        for(int i = 0; i < mgr.getAuthorsBooks().size(); i++)
{
            String[] book = mgr.getAuthorsBooks().get(i);
            Label newTitle = new Label(book[0]);
            Label newSeries = new Label(book[1]);
            Label newVolume = new Label(book[2]);
            Label newISBN = new Label(book[3]);
            Label newPublisher = new Label(book[4]);
            authorPane.add(newTitle, 0, j);
            authorPane.add(newSeries, 1, j);
            authorPane.add(newVolume, 2, j);
            authorPane.add(newISBN, 3, j);
            authorPane.add(newPublisher, 4, j);
            j++;
        }
        authorHomeScreen.setCenter(authorPane);
        authorPane.requestFocus();
    }

    public void createDeletePopup() {
        Label goodbye = new Label("We're sad to see you go
:(");
        goodbye.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 24));
        goodbye.setLayoutY(0);

goodbye.layoutXProperty().bind(deleteProfilePopup.widthPropert
y().subtract(goodbye.widthProperty()).divide(2));
        deleteProfilePopup.setAnchorX(575);
        deleteProfilePopup.setAnchorY(325);
        authorHomeScreen.setCenter(null);
        readerHomeScreen.setCenter(null);
        deleteProfilePopup.getContent().addAll(goodbye);
        if (user == 1) {
            Label authorText = new Label("Please note that any
books already added to Book Nook will not be deleted.");

authorText.layoutXProperty().bind(deleteProfilePopup.widthProp
erty().subtract(authorText.widthProperty()).divide(2));
            authorText.setLayoutY(goodbye.getLayoutY() + 35);
            deleteProfilePopup.getContent().add(authorText);
        }

        Label confirm = new Label("Are you sure you want to
delete your profile?");
```

```
confirm.layoutXProperty().bind(deleteProfilePopup.widthPropert
y().subtract(confirm.widthProperty()).divide(2));
        confirm.setLayoutY(goodbye.getLayoutY() + 100);
        deleteProfilePopup.getContent().add(confirm);

        if (user != 1) {

goodbye.layoutXProperty().bind(deleteProfilePopup.widthPropert
y().subtract(goodbye.widthProperty()).divide(1.01));

confirm.layoutXProperty().bind(deleteProfilePopup.widthPropert
y().subtract(confirm.widthProperty()).divide(1.01));
        }

        Button delete = new Button("Delete Profile");
        delete.setLayoutY(confirm.getLayoutY() + 50);
        delete.setLayoutX(85);

        Button cancel = new Button("Keep Profile");
        cancel.setLayoutY(delete.getLayoutY());
        cancel.setLayoutX(delete.getLayoutX() + 150);

        deleteProfilePopup.getContent().addAll(delete,
cancel);

        cancel.setOnAction(e -> {
            if(user == 1) {
                deleteProfilePopup.hide();
                removeBooks();
                printBooks();
                authorHomeScreen.setCenter(authorPane);
                authorPane.requestFocus();
            } else {
                deleteProfilePopup.hide();
                removeBooks();
                printReadBooks();
                readerHomeScreen.setCenter(readerPane);
                readerPane.requestFocus();
            }

        });

        delete.setOnAction(e2 -> {
            deleteProfilePopup.hide();
            mgr.deleteProfile(username, user);
            loginPane.getUsername().setText("Username");
            loginPane.getPassword().setText("Password");
            if (user == 1) {
                authorHomeScreen.setCenter(loginPane);

                authorHomeScreen.setTop(null);
                removeBooks();
            } else {
                readerHomeScreen.setCenter(loginPane);

                readerHomeScreen.setTop(null);

                removeBooks();
            }
            loginPane.requestFocus();
            Alert deleted = new
Alert(Alert.AlertType.INFORMATION);
            deleted.setHeaderText("Profile Deleted");
            deleted.setContentText("Your profile was
successfully deleted.");
            deleted.show();
        });
    }

    public void printReadBooks() {
        Label name = new Label(mgr.getNameOfUser() + "'s Read
Books");
        name.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 24));
        readerPane.setStyle("-fx-background-color:
antiquewhite;");
        readerPane.add(name, 0, 0);
        Label title = new Label("Title");
        title.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label author = new Label("Author");
        author.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label series = new Label("Series");
        series.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label volume = new Label("Volume");
        volume.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label isbn = new Label("ISBN");
        isbn.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label publisher = new Label("Publisher");
        publisher.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label blank = new Label("");
        readerPane.add(blank, 0, 1);
        readerPane.setHgap(100);
        readerPane.add(title, 0,2);
        readerPane.add(author, 1, 2);
        readerPane.add(series, 2,2);
        readerPane.add(volume, 3,2);
        readerPane.add(isbn, 4,2);
        readerPane.add(publisher, 5,2);
        mgr.pullReadBooks(username);
        int j = 3;
        for(int i = 0; i < mgr.getReadBooks().size(); i++) {
            String[] book = mgr.getReadBooks().get(i);
            Label newTitle = new Label(book[0]);
            Label newAuthor = new Label(book[1]);
            Label newSeries = new Label(book[2]);
            Label newVolume = new Label(book[3]);
            Label newISBN = new Label(book[4]);
            Label newPublisher = new Label(book[5]);
            readerPane.add(newTitle, 0, j);
            readerPane.add(newAuthor, 1, j);
            readerPane.add(newSeries, 2, j);
            readerPane.add(newVolume, 3, j);
            readerPane.add(newISBN, 4, j);
            readerPane.add(newPublisher, 5, j);
            j++;
        }
        readerHomeScreen.setCenter(readerPane);
        readerPane.requestFocus();
    }

    public void printTBR() {
        Label name = new Label(mgr.getNameOfUser() + "'s
TBR");
        name.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 24));
        readerPane.setStyle("-fx-background-color:
antiquewhite;");
        readerPane.add(name, 0, 0);
        Label title = new Label("Title");
        title.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label author = new Label("Author");
        author.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label series = new Label("Series");
        series.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label volume = new Label("Volume");
        volume.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label isbn = new Label("ISBN");
        isbn.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label publisher = new Label("Publisher");
        publisher.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label blank = new Label("");
        readerPane.add(blank, 0, 1);
        readerPane.setHgap(100);
        readerPane.add(title, 0,2);
        readerPane.add(author, 1, 2);
        readerPane.add(series, 2,2);
        readerPane.add(volume, 3,2);
        readerPane.add(isbn, 4,2);
        readerPane.add(publisher, 5,2);
        mgr.pullTBR(username);
        int j = 3;
        for(int i = 0; i < mgr.getTBRBooks().size(); i++) {
            String[] book = mgr.getTBRBooks().get(i);
            Label newTitle = new Label(book[0]);
            Label newAuthor = new Label(book[1]);
            Label newSeries = new Label(book[2]);
            Label newVolume = new Label(book[3]);
            Label newISBN = new Label(book[4]);
            Label newPublisher = new Label(book[5]);
            readerPane.add(newTitle, 0, j);
            readerPane.add(newAuthor, 1, j);
            readerPane.add(newSeries, 2, j);
            readerPane.add(newVolume, 3, j);
            readerPane.add(newISBN, 4, j);
            readerPane.add(newPublisher, 5, j);
            j++;
        }
        readerHomeScreen.setCenter(readerPane);
        readerPane.requestFocus();
    }

    public void printReadingBooks() {
        Label name = new Label(mgr.getNameOfUser() + "'s
Currently Reading");
        name.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 24));
        readerPane.setStyle("-fx-background-color:
antiquewhite;");
        readerPane.add(name, 0, 0);
        Label title = new Label("Title");
        title.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label author = new Label("Author");
        author.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label series = new Label("Series");
        series.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label volume = new Label("Volume");
        volume.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label isbn = new Label("ISBN");
        isbn.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label publisher = new Label("Publisher");
        publisher.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label blank = new Label("");
        readerPane.add(blank, 0, 1);
        readerPane.setHgap(100);
        readerPane.add(title, 0,2);
        readerPane.add(author, 1, 2);
        readerPane.add(series, 2,2);
        readerPane.add(volume, 3,2);
        readerPane.add(isbn, 4,2);
        readerPane.add(publisher, 5,2);
        mgr.pullReadingBooks(username);
        int j = 3;
        for(int i = 0; i < mgr.getReadingBooks().size(); i++)
{
            String[] book = mgr.getReadingBooks().get(i);
            Label newTitle = new Label(book[0]);
            Label newAuthor = new Label(book[1]);
            Label newSeries = new Label(book[2]);
            Label newVolume = new Label(book[3]);
            Label newISBN = new Label(book[4]);
            Label newPublisher = new Label(book[5]);
            readerPane.add(newTitle, 0, j);
            readerPane.add(newAuthor, 1, j);
            readerPane.add(newSeries, 2, j);
            readerPane.add(newVolume, 3, j);
            readerPane.add(newISBN, 4, j);
            readerPane.add(newPublisher, 5, j);
            j++;
        }
        readerHomeScreen.setCenter(readerPane);
        readerPane.requestFocus();
    }

    public void searchBooks() {
        readingStatusISBN = null;
        String search = bookSearchPane.getSearch().getText();
        mgr.pullSearch(search);
        GridPane pane = new GridPane();
        pane.getChildren().clear();
        pane.setStyle("-fx-background-color: antiquewhite;");
        Label title = new Label("Title");
        title.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label author = new Label("Author");
        author.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label series = new Label("Series");
        series.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label volume = new Label("Volume");
        volume.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label isbn = new Label("ISBN");
        isbn.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
```

```
        Label publisher = new Label("Publisher");
        publisher.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label synopsis = new Label("Synopsis");
        synopsis.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label readingStatus = new Label("Reading Status");
        readingStatus.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        Label blank = new Label("");
        Label blank2 = new Label("");
        pane.add(blank, 0, 0);
        pane.add(blank2, 0, 2);
        pane.setHgap(75);
        pane.setVgap(50);
        pane.add(title, 0,1);
        pane.add(author, 1, 1);
        pane.add(series, 2,1);
        pane.add(volume, 3,1);
        pane.add(isbn, 4,1);
        pane.add(publisher, 5,1);
        pane.add(synopsis, 6, 1);
        pane.add(readingStatus, 7, 1);
        int k = 1;
        int j = 2;
        Button[] buttons = new Button[mgr.getSearch().size()];
        for(int i = 0; i < mgr.getSearch().size(); i++) {
            String[] result = mgr.getSearch().get(i);
            Label newTitle = new Label(result[0]);
            Label newAuthor = new Label(result[1]);
            Label newSeries = new Label(result[2]);
            Label newVolume = new Label(result[3]);
            Label newISBN = new Label(result[4]);
            Label newPublisher = new Label(result[5]);
            TextArea newSynopsis = new TextArea();
            newSynopsis.setPrefWidth(400);
            newSynopsis.setText(result[6]);
            newSynopsis.setWrapText(true);
            newSynopsis.setEditable(false);
            Button addReadingStatus = new Button("Select
Reading Status");
            buttons[i] = addReadingStatus;
            pane.add(newTitle, 0, j);
            pane.add(newAuthor, 1, j);
            pane.add(newSeries, 2, j);
            pane.add(newVolume, 3, j);
            pane.add(newISBN, 4, j);
            pane.add(newPublisher, 5, j);
            pane.add(newSynopsis, 6, j);
            pane.add(buttons[i], 7, j);
            j++;
        }
        bookSearchPane.setCenter(pane);
        pane.requestFocus();

        for(int i = 0; i < buttons.length; i++) {
            int finalI = i;
            buttons[i].setOnAction(e ->{
                String[] book = mgr.getSearch().get(finalI);
                readingStatusISBN = book[4];
                mgr.getSearch().clear();
                readerHomeScreen.setCenter(readingStatusPane);

readingStatusPane.getISBN().setText(readingStatusISBN);
                readingStatusPane.requestFocus();
                bookSearchPane.getSearch().setText("");
                pane.getChildren().clear();

readingStatusPane.getReadingStatus().setValue("");

readingStatusPane.getSearch().setVisible(true);

readingStatusPane.getSetReadingStatus().setVisible(false);

readingStatusPane.getReadingStatus().setVisible(false);

readingStatusPane.getSetReadingStatusText().setVisible(false);

readingStatusPane.getEditReadingStatus().setVisible(false);

readingStatusPane.getRemoveStatus().setVisible(false);
            });

        }
    }

    public void createReviewPane() {
        Pane pane = new Pane();
        pane.setStyle("-fx-background-color: antiquewhite;");
        mgr.recallBook(readingStatusISBN);
        Label name = new Label(mgr.getNameOfUser() + "'s 
Review of " + mgr.getBookDetails()[0] + ":");
        name.setFont(Font.font("Times New Roman", 
FontWeight.BOLD, 24));
        name.setLayoutY(10);
        name.setLayoutX(10);
        Label zeroText = new Label("Please enter 0 if you 
choose not to review or category is not applicable.");
        zeroText.setFont(Font.font("Times New Roman", 
FontWeight.BOLD, 14));
        zeroText.setLayoutY(name.getLayoutY() + 25);
        zeroText.setLayoutX(10);
        pane.getChildren().addAll(name, zeroText);
        Label storylineText = new Label("Storyline: ");
        storylineText.setLayoutY(zeroText.getLayoutY() + 50);
        storylineText.setLayoutX(10);
        TextField storyline = new TextField();
        storyline.setLayoutX(storylineText.getLayoutX() + 70);
        storyline.setLayoutY(storylineText.getLayoutY() - 3);
        storyline.setPrefWidth(50);
        storyline.setAlignment(Pos.CENTER);
        Label storylineFraction = new Label("/ 5.00");
        storylineFraction.setLayoutX(storyline.getLayoutX() +
60);

storylineFraction.setLayoutY(storylineText.getLayoutY());
        pane.getChildren().addAll(storylineText, storyline,
storylineFraction);

        Label plotText = new Label("Plot: ");
        plotText.setLayoutY(storylineText.getLayoutY() + 50);
        plotText.setLayoutX(10);
        TextField plot = new TextField();
        plot.setLayoutX(plotText.getLayoutX() + 70);
        plot.setLayoutY(plotText.getLayoutY() - 3);
        plot.setPrefWidth(50);
        plot.setAlignment(Pos.CENTER);
        Label plotFraction = new Label("/ 5.00");
        plotFraction.setLayoutX(plot.getLayoutX() + 60);
        plotFraction.setLayoutY(plotText.getLayoutY());
        pane.getChildren().addAll(plotText, plot,
plotFraction);

        Label settingText = new Label("Setting: ");
        settingText.setLayoutY(plotText.getLayoutY() + 50);
        settingText.setLayoutX(10);
        TextField setting = new TextField();
        setting.setLayoutX(settingText.getLayoutX() + 70);
        setting.setLayoutY(settingText.getLayoutY() - 3);
        setting.setPrefWidth(50);
        setting.setAlignment(Pos.CENTER);
        Label settingFraction = new Label("/ 5.00");
        settingFraction.setLayoutX(setting.getLayoutX() + 60);
        settingFraction.setLayoutY(settingText.getLayoutY());
        pane.getChildren().addAll(settingText, setting,
settingFraction);

        Label spiceText = new Label("Spice: ");
        spiceText.setLayoutY(settingText.getLayoutY() + 50);
        spiceText.setLayoutX(10);
        TextField spice = new TextField();
        spice.setLayoutX(spiceText.getLayoutX() + 70);
        spice.setLayoutY(spiceText.getLayoutY() - 3);
        spice.setPrefWidth(50);
        spice.setAlignment(Pos.CENTER);
        Label spiceFraction = new Label("/ 5.00");
        spiceFraction.setLayoutX(spice.getLayoutX() + 60);
        spiceFraction.setLayoutY(spiceText.getLayoutY());
        pane.getChildren().addAll(spiceText, spice,
spiceFraction);

        Label charactersText = new Label("Characters: ");
        charactersText.setLayoutY(spiceText.getLayoutY() +
50);
        charactersText.setLayoutX(10);
        TextField characters = new TextField();
        characters.setLayoutX(charactersText.getLayoutX() +
70);
        characters.setLayoutY(charactersText.getLayoutY() -
3);
        characters.setPrefWidth(50);
        characters.setAlignment(Pos.CENTER);
        Label charactersFraction = new Label("/ 5.00");
        charactersFraction.setLayoutX(characters.getLayoutX()
+ 60);

charactersFraction.setLayoutY(charactersText.getLayoutY());
        pane.getChildren().addAll(charactersText, characters,
charactersFraction);

        Label worldBuildingText = new Label("World Building:
");

worldBuildingText.setLayoutY(charactersText.getLayoutY() +
50);
        worldBuildingText.setLayoutX(10);
        TextField worldBuilding = new TextField();

worldBuilding.setLayoutX(worldBuildingText.getLayoutX() + 90);

worldBuilding.setLayoutY(worldBuildingText.getLayoutY() - 3);
        worldBuilding.setPrefWidth(50);
        worldBuilding.setAlignment(Pos.CENTER);
        Label worldBuildingFraction = new Label("/ 5.00");

worldBuildingFraction.setLayoutX(worldBuilding.getLayoutX() +
60);

worldBuildingFraction.setLayoutY(worldBuildingText.getLayoutY(
));
        pane.getChildren().addAll(worldBuildingText,
worldBuilding, worldBuildingFraction);

        Label writingStyleText = new Label("Writing Style: ");

writingStyleText.setLayoutY(worldBuildingText.getLayoutY() +
50);
        writingStyleText.setLayoutX(10);
        TextField writingStyle = new TextField();
        writingStyle.setLayoutX(writingStyleText.getLayoutX()
+ 90);
        writingStyle.setLayoutY(writingStyleText.getLayoutY()
- 3);
        writingStyle.setPrefWidth(50);
        writingStyle.setAlignment(Pos.CENTER);
        Label writingStyleFraction = new Label("/ 5.00");

writingStyleFraction.setLayoutX(writingStyle.getLayoutX() +
60);

writingStyleFraction.setLayoutY(writingStyleText.getLayoutY())
;
        pane.getChildren().addAll(writingStyleText,
writingStyle, writingStyleFraction);

        Label dateInstructions = new Label("Please enter a two
digit month, two digit day, and four digit year.");
        dateInstructions.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        dateInstructions.setLayoutX(10);

dateInstructions.setLayoutY(writingStyleText.getLayoutY() +
50);
        pane.getChildren().add(dateInstructions);

        Label dateStartedText = new Label("Date Started: ");

dateStartedText.setLayoutY(dateInstructions.getLayoutY() +
50);
        dateStartedText.setLayoutX(10);
        TextField dateStartedMonth = new TextField();

dateStartedMonth.setLayoutX(dateStartedText.getLayoutX() +
90);

dateStartedMonth.setLayoutY(dateStartedText.getLayoutY() - 3);
        dateStartedMonth.setPrefWidth(30);
        dateStartedMonth.setAlignment(Pos.CENTER);
        Label date1 = new Label("/ ");
        date1.setLayoutX(dateStartedMonth.getLayoutX() + 45);
        date1.setLayoutY(dateStartedText.getLayoutY());
        TextField dateStartedDay = new TextField();

dateStartedDay.setLayoutX(dateStartedMonth.getLayoutX() + 60);
        dateStartedDay.setLayoutY(dateStartedText.getLayoutY()
- 3);
        dateStartedDay.setPrefWidth(30);
        dateStartedDay.setAlignment(Pos.CENTER);
        Label date2 = new Label("/ ");
        date2.setLayoutX(dateStartedDay.getLayoutX() + 45);
        date2.setLayoutY(dateStartedText.getLayoutY());
        TextField dateStartedYear = new TextField();
        dateStartedYear.setLayoutX(dateStartedDay.getLayoutX()
+ 60);

dateStartedYear.setLayoutY(dateStartedText.getLayoutY() - 3);
        dateStartedYear.setPrefWidth(50);
        dateStartedYear.setAlignment(Pos.CENTER);
        pane.getChildren().addAll(dateStartedText,
dateStartedMonth, date1, dateStartedDay, date2,
dateStartedYear);

        Label dateFinishedText = new Label("Date Finished: ");

dateFinishedText.setLayoutY(dateStartedText.getLayoutY() +
50);
        dateFinishedText.setLayoutX(10);
        TextField dateFinishedMonth = new TextField();

dateFinishedMonth.setLayoutX(dateFinishedText.getLayoutX() +
90);

dateFinishedMonth.setLayoutY(dateFinishedText.getLayoutY() -
3);
        dateFinishedMonth.setPrefWidth(30);
        dateFinishedMonth.setAlignment(Pos.CENTER);
        Label date3 = new Label("/ ");
        date3.setLayoutX(dateStartedMonth.getLayoutX() + 45);
        date3.setLayoutY(dateStartedText.getLayoutY());
        TextField dateFinishedDay = new TextField();

dateFinishedDay.setLayoutX(dateFinishedMonth.getLayoutX() +
60);

dateFinishedDay.setLayoutY(dateFinishedText.getLayoutY() - 3);
        dateFinishedDay.setPrefWidth(30);
        dateFinishedDay.setAlignment(Pos.CENTER);
        Label date4 = new Label("/ ");
        date4.setLayoutX(dateFinishedDay.getLayoutX() + 45);
        date4.setLayoutY(dateFinishedText.getLayoutY());
        TextField dateFinishedYear = new TextField();

dateFinishedYear.setLayoutX(dateFinishedDay.getLayoutX() +
60);

dateFinishedYear.setLayoutY(dateFinishedText.getLayoutY() -
3);
        dateFinishedYear.setPrefWidth(50);
        dateFinishedYear.setAlignment(Pos.CENTER);
        pane.getChildren().addAll(dateFinishedText,
dateFinishedMonth, date3, dateFinishedDay, date4,
dateFinishedYear);

        Label thoughtsText = new Label("Thoughts:");
        thoughtsText.setLayoutY(settingText.getLayoutY());
        thoughtsText.setLayoutX(settingText.getLayoutX() +
600);
        TextArea thoughts = new TextArea();
        thoughts.setWrapText(true);
        thoughts.setPrefHeight(150);
        thoughts.setPrefWidth(500);
        thoughts.setLayoutX(thoughtsText.getLayoutX());
        thoughts.setLayoutY(thoughtsText.getLayoutY() + 20);
        pane.getChildren().addAll(thoughtsText, thoughts);

        Button createReview = new Button("Done!");

createReview.setLayoutY(dateFinishedText.getLayoutY());
        createReview.setLayoutX(thoughtsText.getLayoutX());
        pane.getChildren().add(createReview);

        createReview.setOnAction(e -> {
            Float storylineRating =
Float.parseFloat(storyline.getText());
            Float plotRating =
Float.parseFloat(plot.getText());
            Float settingRating =
Float.parseFloat(setting.getText());
            Float spiceRating =
Float.parseFloat(spice.getText());
            Float characterRating =
Float.parseFloat(characters.getText());
            Float wbRating =
Float.parseFloat(worldBuilding.getText());
            Float wsRating =
Float.parseFloat(writingStyle.getText());
            String startedMonth = dateStartedMonth.getText();
            String startedDay = dateStartedDay.getText();
            String startedYear = dateStartedYear.getText();
            String dateStarted = startedYear + "-" +
startedMonth + "-" + startedDay;
            String finishedMonth =
dateFinishedMonth.getText();
            String finishedDay = dateFinishedDay.getText();
            String finishedYear = dateFinishedYear.getText();
            String dateFinished = finishedYear + "-" +
finishedMonth + "-" + finishedDay;
            String writtenReview;
            if(thoughts.getText().equals(null)) {
                writtenReview = null;
            } else {
                writtenReview = thoughts.getText();
            }
            mgr.createReview(username, readingStatusISBN,
storylineRating, plotRating, settingRating, spiceRating,
characterRating, wbRating, wsRating, dateStarted,
dateFinished, writtenReview);
            removeBooks();
            printReadBooks();
            readerHomeScreen.setCenter(readerPane);
            readerPane.requestFocus();
        });

        readerHomeScreen.setCenter(pane);
        pane.requestFocus();
    }

    public void updateReview(String isbn) {
        Pane pane = new Pane();
        pane.setStyle("-fx-background-color: antiquewhite;");
        mgr.recallBook(isbn);
        Label name = new Label(mgr.getNameOfUser() + "'s
Review of " + mgr.getBookDetails()[0] + ":");
        name.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 24));
        name.setLayoutY(10);
        name.setLayoutX(10);
        Label zeroText = new Label("Please enter 0 if you
choose not to review or category is not applicable.");
        zeroText.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        zeroText.setLayoutY(name.getLayoutY() + 25);
        zeroText.setLayoutX(10);
        pane.getChildren().addAll(name, zeroText);

        Label storylineText = new Label("Storyline: ");
        storylineText.setLayoutY(zeroText.getLayoutY() + 50);
        storylineText.setLayoutX(10);
        TextField storyline = new TextField();
        storyline.setLayoutX(storylineText.getLayoutX() + 70);
        storyline.setLayoutY(storylineText.getLayoutY() - 3);
        storyline.setPrefWidth(50);
        storyline.setAlignment(Pos.CENTER);
        Label storylineFraction = new Label("/ 5.00");
        storylineFraction.setLayoutX(storyline.getLayoutX() +
60);

storylineFraction.setLayoutY(storylineText.getLayoutY());
        pane.getChildren().addAll(storylineText, storyline,
storylineFraction);

        Label plotText = new Label("Plot: ");
        plotText.setLayoutY(storylineText.getLayoutY() + 50);
        plotText.setLayoutX(10);
        TextField plot = new TextField();
        plot.setLayoutX(plotText.getLayoutX() + 70);
        plot.setLayoutY(plotText.getLayoutY() - 3);
        plot.setPrefWidth(50);
        plot.setAlignment(Pos.CENTER);
        Label plotFraction = new Label("/ 5.00");
        plotFraction.setLayoutX(plot.getLayoutX() + 60);
        plotFraction.setLayoutY(plotText.getLayoutY());
        pane.getChildren().addAll(plotText, plot,
plotFraction);

        Label settingText = new Label("Setting: ");
        settingText.setLayoutY(plotText.getLayoutY() + 50);
        settingText.setLayoutX(10);
        TextField setting = new TextField();
        setting.setLayoutX(settingText.getLayoutX() + 70);
        setting.setLayoutY(settingText.getLayoutY() - 3);
        setting.setPrefWidth(50);
        setting.setAlignment(Pos.CENTER);
        Label settingFraction = new Label("/ 5.00");
        settingFraction.setLayoutX(setting.getLayoutX() + 60);
        settingFraction.setLayoutY(settingText.getLayoutY());
        pane.getChildren().addAll(settingText, setting,
settingFraction);

        Label spiceText = new Label("Spice: ");
        spiceText.setLayoutY(settingText.getLayoutY() + 50);
        spiceText.setLayoutX(10);
        TextField spice = new TextField();
        spice.setLayoutX(spiceText.getLayoutX() + 70);
        spice.setLayoutY(spiceText.getLayoutY() - 3);
        spice.setPrefWidth(50);
        spice.setAlignment(Pos.CENTER);
        Label spiceFraction = new Label("/ 5.00");
        spiceFraction.setLayoutX(spice.getLayoutX() + 60);
        spiceFraction.setLayoutY(spiceText.getLayoutY());
        pane.getChildren().addAll(spiceText, spice,
spiceFraction);

        Label charactersText = new Label("Characters: ");
        charactersText.setLayoutY(spiceText.getLayoutY() +
50);
        charactersText.setLayoutX(10);
        TextField characters = new TextField();
        characters.setLayoutX(charactersText.getLayoutX() +
70);
        characters.setLayoutY(charactersText.getLayoutY() -
3);
        characters.setPrefWidth(50);
        characters.setAlignment(Pos.CENTER);
        Label charactersFraction = new Label("/ 5.00");
        charactersFraction.setLayoutX(characters.getLayoutX()
+ 60);

charactersFraction.setLayoutY(charactersText.getLayoutY());
        pane.getChildren().addAll(charactersText, characters,
charactersFraction);

        Label worldBuildingText = new Label("World Building:
");
```

```
worldBuildingText.setLayoutY(charactersText.getLayoutY() +
50);
        worldBuildingText.setLayoutX(10);
        TextField worldBuilding = new TextField();

worldBuilding.setLayoutX(worldBuildingText.getLayoutX() + 90);

worldBuilding.setLayoutY(worldBuildingText.getLayoutY() - 3);
        worldBuilding.setPrefWidth(50);
        worldBuilding.setAlignment(Pos.CENTER);
        Label worldBuildingFraction = new Label("/ 5.00");

worldBuildingFraction.setLayoutX(worldBuilding.getLayoutX() +
60);

worldBuildingFraction.setLayoutY(worldBuildingText.getLayoutY(
));
        pane.getChildren().addAll(worldBuildingText,
worldBuilding, worldBuildingFraction);

        Label writingStyleText = new Label("Writing Style: ");

writingStyleText.setLayoutY(worldBuildingText.getLayoutY() +
50);
        writingStyleText.setLayoutX(10);
        TextField writingStyle = new TextField();
        writingStyle.setLayoutX(writingStyleText.getLayoutX()
+ 90);
        writingStyle.setLayoutY(writingStyleText.getLayoutY()
- 3);
        writingStyle.setPrefWidth(50);
        writingStyle.setAlignment(Pos.CENTER);
        Label writingStyleFraction = new Label("/ 5.00");

writingStyleFraction.setLayoutX(writingStyle.getLayoutX() +
60);

writingStyleFraction.setLayoutY(writingStyleText.getLayoutY())
;
        pane.getChildren().addAll(writingStyleText,
writingStyle, writingStyleFraction);

        Label dateInstructions = new Label("Please enter a two
digit month, two digit day, and four digit year.");
        dateInstructions.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));
        dateInstructions.setLayoutX(10);

dateInstructions.setLayoutY(writingStyleText.getLayoutY() +
50);
        pane.getChildren().add(dateInstructions);
        Label dateStartedText = new Label("Date Started: ");

dateStartedText.setLayoutY(dateInstructions.getLayoutY() +
50);
        dateStartedText.setLayoutX(10);
        TextField dateStartedMonth = new TextField();

dateStartedMonth.setLayoutX(dateStartedText.getLayoutX() +
90);

dateStartedMonth.setLayoutY(dateStartedText.getLayoutY() - 3);
        dateStartedMonth.setPrefWidth(30);
        dateStartedMonth.setAlignment(Pos.CENTER);
        Label date1 = new Label("/ ");
        date1.setLayoutX(dateStartedMonth.getLayoutX() + 45);
        date1.setLayoutY(dateStartedText.getLayoutY());
        TextField dateStartedDay = new TextField();

dateStartedDay.setLayoutX(dateStartedMonth.getLayoutX() + 60);
        dateStartedDay.setLayoutY(dateStartedText.getLayoutY()
- 3);
        dateStartedDay.setPrefWidth(30);
        dateStartedDay.setAlignment(Pos.CENTER);
        Label date2 = new Label("/ ");
        date2.setLayoutX(dateStartedDay.getLayoutX() + 45);
        date2.setLayoutY(dateStartedText.getLayoutY());
        TextField dateStartedYear = new TextField();
        dateStartedYear.setLayoutX(dateStartedDay.getLayoutX()
+ 60);

dateStartedYear.setLayoutY(dateStartedText.getLayoutY() - 3);
        dateStartedYear.setPrefWidth(50);
        dateStartedYear.setAlignment(Pos.CENTER);
        pane.getChildren().addAll(dateStartedText,
dateStartedMonth, date1, dateStartedDay, date2,
dateStartedYear);

        Label dateFinishedText = new Label("Date Finished: ");

dateFinishedText.setLayoutY(dateStartedText.getLayoutY() +
50);
        dateFinishedText.setLayoutX(10);
        TextField dateFinishedMonth = new TextField();

dateFinishedMonth.setLayoutX(dateFinishedText.getLayoutX() +
90);

dateFinishedMonth.setLayoutY(dateFinishedText.getLayoutY() -
3);
        dateFinishedMonth.setPrefWidth(30);
        dateFinishedMonth.setAlignment(Pos.CENTER);
        Label date3 = new Label("/ ");
        date3.setLayoutX(dateStartedMonth.getLayoutX() + 45);
        date3.setLayoutY(dateStartedText.getLayoutY());
        TextField dateFinishedDay = new TextField();

dateFinishedDay.setLayoutX(dateFinishedMonth.getLayoutX() +
60);

dateFinishedDay.setLayoutY(dateFinishedText.getLayoutY() - 3);
        dateFinishedDay.setPrefWidth(30);
        dateFinishedDay.setAlignment(Pos.CENTER);
        Label date4 = new Label("/ ");
        date4.setLayoutX(dateFinishedDay.getLayoutX() + 45);
        date4.setLayoutY(dateFinishedText.getLayoutY());
        TextField dateFinishedYear = new TextField();

dateFinishedYear.setLayoutX(dateFinishedDay.getLayoutX() +
60);

dateFinishedYear.setLayoutY(dateFinishedText.getLayoutY() -
3);
        dateFinishedYear.setPrefWidth(50);
        dateFinishedYear.setAlignment(Pos.CENTER);
        pane.getChildren().addAll(dateFinishedText,
dateFinishedMonth, date3, dateFinishedDay, date4,
dateFinishedYear);

        Label thoughtsText = new Label("Thoughts:");
        thoughtsText.setLayoutY(settingText.getLayoutY());
        thoughtsText.setLayoutX(settingText.getLayoutX() +
600);
        TextArea thoughts = new TextArea();
        thoughts.setWrapText(true);
        thoughts.setPrefHeight(150);
        thoughts.setPrefWidth(500);
        thoughts.setLayoutX(thoughtsText.getLayoutX());
        thoughts.setLayoutY(thoughtsText.getLayoutY() + 20);
        pane.getChildren().addAll(thoughtsText, thoughts);

        Button updateReview = new Button("Done!");

updateReview.setLayoutY(dateFinishedText.getLayoutY());
        updateReview.setLayoutX(thoughtsText.getLayoutX());
        pane.getChildren().add(updateReview);

        String[] review = mgr.pullReviewDetails(username,
isbn);
        storyline.setText(review[0]);
        plot.setText(review[1]);
        setting.setText(review[2]);
        spice.setText(review[3]);
        characters.setText(review[4]);
        worldBuilding.setText(review[5]);
        writingStyle.setText(review[6]);
        String[] dateStarted = review[7].split("-");
        dateStartedYear.setText(dateStarted[0]);
        dateStartedMonth.setText(dateStarted[1]);
        dateStartedDay.setText(dateStarted[2]);
        String[] dateFinished = review[8].split("-");
        dateFinishedYear.setText(dateFinished[0]);
        dateFinishedMonth.setText(dateFinished[1]);
        dateFinishedDay.setText(dateFinished[2]);
        thoughts.setText(review[9]);

        updateReview.setOnAction(e -> {
            Float storylineRating =
Float.parseFloat(storyline.getText());
            Float plotRating =
Float.parseFloat(plot.getText());
            Float settingRating =
Float.parseFloat(setting.getText());
            Float spiceRating =
Float.parseFloat(spice.getText());
            Float characterRating =
Float.parseFloat(characters.getText());
            Float wbRating =
Float.parseFloat(worldBuilding.getText());
            Float wsRating =
Float.parseFloat(writingStyle.getText());
            String startedMonth = dateStartedMonth.getText();
            String startedDay = dateStartedDay.getText();
            String startedYear = dateStartedYear.getText();
            String newDateStarted = startedYear + "-" +
startedMonth + "-" + startedDay;
            String finishedMonth =
dateFinishedMonth.getText();
            String finishedDay = dateFinishedDay.getText();
            String finishedYear = dateFinishedYear.getText();
            String newDateFinished = finishedYear + "-" +
finishedMonth + "-" + finishedDay;
            String writtenReview;
            writtenReview = thoughts.getText();
            mgr.updateReview(username, isbn, storylineRating,
plotRating, settingRating, spiceRating, characterRating,
wbRating, wsRating, newDateStarted, newDateFinished,
writtenReview);
            removeBooks();
            printReadBooks();
            readerHomeScreen.setCenter(readerPane);
            readerPane.requestFocus();
        });

        readerHomeScreen.setCenter(pane);
        pane.requestFocus();
    }

    public void createUpdate() {
        updatePopup.setHeight(200);
        updatePopup.setWidth(200);
        updatePopup.setAnchorX(575);
        updatePopup.setAnchorY(325);
        if(user == 1) {
            Label text = new Label("Please enter the ISBN of
the book to update:");
            TextField ISBN = new TextField();
            Button update = new Button("Update");
            update.setLayoutX(ISBN.getLayoutX() + 148);
            update.setLayoutY(ISBN.getLayoutY() + 100);
            text.setLayoutX(65);
            text.setLayoutY(0);
            ISBN.setLayoutX(100);
            ISBN.setLayoutY(text.getLayoutY() + 50);
            updatePopup.getContent().addAll(text, ISBN,
update);
            update.setOnAction(e -> {
                if (ISBN.getText().length() != 13) {
                    Alert error = new
Alert(Alert.AlertType.ERROR);
                    error.setTitle("Uh oh...");
                    error.setHeaderText("Invalid ISBN");
                    error.setContentText("Please enter a valid
ISBN. They are strings of 13 numbers.");
                    error.show();
                } else {
                    String isbn = ISBN.getText();
                    bookUpdatePane.setISBN(isbn);
                    updatePopup.hide();
                    mgr.recallBook(isbn);
                    fillBoxes();

authorHomeScreen.setCenter(bookUpdatePane);
                    ISBN.setText("");
                }
            });
        } else {
            Label text = new Label("Please enter the ISBN of
the book to update:");
            TextField ISBN = new TextField();
            Button update = new Button("Update");
            update.setLayoutX(ISBN.getLayoutX() + 148);
            update.setLayoutY(ISBN.getLayoutY() + 100);
            text.setLayoutX(65);
            text.setLayoutY(0);
            ISBN.setLayoutX(100);
            ISBN.setLayoutY(text.getLayoutY() + 50);
            updatePopup.getContent().addAll(text, ISBN,
update);
            update.setOnAction(e -> {
                if (ISBN.getText().length() != 13) {
                    Alert error = new
Alert(Alert.AlertType.ERROR);
```

```
                    error.setTitle("Uh oh...");
                    error.setHeaderText("Invalid ISBN");
                    error.setContentText("Please enter a valid
ISBN. They are strings of 13 numbers.");
                    error.show();
                    error.setY(500);
                } else {
                    String isbn = ISBN.getText();
                    updatePopup.hide();
                    updateReview(isbn);
                    ISBN.setText("");
                }
            });
        }
    }

    public void createCharts(ArrayList reviews) {
        int[] storyline = (int[]) reviews.get(0);
        int[] plot = (int[]) reviews.get(1);
        int[] setting = (int[]) reviews.get(2);
        int[] spice = (int[]) reviews.get(3);
        int[] characters = (int[]) reviews.get(4);
        int[] wb = (int[]) reviews.get(5);
        int[] ws = (int[]) reviews.get(6);


        ObservableList<PieChart.Data> storylineData =
                FXCollections.observableArrayList(
                        new PieChart.Data("0", storyline[0]),
                        new PieChart.Data("1", storyline[1]),
                        new PieChart.Data("2", storyline[2]),
                        new PieChart.Data("3", storyline[3]),
                        new PieChart.Data("4", storyline[4]),
                        new PieChart.Data("5", storyline[5]));
        storylineChart.setMaxSize(200, 200);
        storylineChart.setData(storylineData);
        storylineChart.setClockwise(true);
        storylineChart.setLabelsVisible(true);
        storylineChart.setLegendSide(Side.RIGHT);
        storylineChart.setLegendVisible(true);
        storylineChart.setLayoutX(10);
        storylineChart.setLayoutY(20);

        ObservableList<PieChart.Data> plotData =
                FXCollections.observableArrayList(
                        new PieChart.Data("0", plot[0]),
                        new PieChart.Data("1", plot[1]),
                        new PieChart.Data("2", plot[2]),
                        new PieChart.Data("3", plot[3]),
                        new PieChart.Data("4", plot[4]),
                        new PieChart.Data("5", plot[5]));
        plotChart.setMaxSize(200, 200);
        plotChart.setData(plotData);
        plotChart.setClockwise(true);
        plotChart.setLabelsVisible(true);
        plotChart.setLegendVisible(true);
        plotChart.setLegendSide(Side.RIGHT);
        plotChart.setLayoutX(storylineChart.getLayoutX());
        plotChart.setLayoutY(storylineChart.getLayoutY() +
210);

        ObservableList<PieChart.Data> spiceData =
                FXCollections.observableArrayList(
                        new PieChart.Data("0", spice[0]),
                        new PieChart.Data("1", spice[1]),
                        new PieChart.Data("2", spice[2]),
                        new PieChart.Data("3", spice[3]),
                        new PieChart.Data("4", spice[4]),
                        new PieChart.Data("5", spice[5]));
        spiceChart.setMaxSize(200, 200);
        spiceChart.setData(spiceData);
        spiceChart.setClockwise(true);
        spiceChart.setLabelsVisible(true);
        spiceChart.setLegendVisible(true);
        spiceChart.setLegendSide(Side.RIGHT);
        spiceChart.setLayoutX(storylineChart.getLayoutX() +
600);
        spiceChart.setLayoutY(storylineChart.getLayoutY());

        ObservableList<PieChart.Data> charactersData =
                FXCollections.observableArrayList(
                        new PieChart.Data("0", characters[0]),
                        new PieChart.Data("1", characters[1]),
                        new PieChart.Data("2", characters[2]),
                        new PieChart.Data("3", characters[3]),
                        new PieChart.Data("4", characters[4]),
                        new PieChart.Data("5",
characters[5]));
        charactersChart.setMaxSize(200, 200);
        charactersChart.setData(charactersData);
        charactersChart.setClockwise(true);
        charactersChart.setLabelsVisible(true);
        charactersChart.setLegendVisible(true);
        charactersChart.setLegendSide(Side.RIGHT);
        charactersChart.setLayoutX(plotChart.getLayoutX() +
600);
        charactersChart.setLayoutY(plotChart.getLayoutY());

        ObservableList<PieChart.Data> settingData =
                FXCollections.observableArrayList(
                        new PieChart.Data("0", setting[0]),
                        new PieChart.Data("1", setting[1]),
                        new PieChart.Data("2", setting[2]),
                        new PieChart.Data("3", setting[3]),
                        new PieChart.Data("4", setting[4]),
                        new PieChart.Data("5", setting[5]));
        settingChart.setMaxSize(200, 200);
        settingChart.setData(settingData);
        settingChart.setClockwise(true);
        settingChart.setLabelsVisible(true);
        settingChart.setLegendVisible(true);
        settingChart.setLegendSide(Side.RIGHT);
        settingChart.setLayoutX(charactersChart.getLayoutX());
        settingChart.setLayoutY(charactersChart.getLayoutY() +
210);

        ObservableList<PieChart.Data> wbData =
                FXCollections.observableArrayList(
                        new PieChart.Data("0", wb[0]),
                        new PieChart.Data("1", wb[1]),
                        new PieChart.Data("2", wb[2]),
                        new PieChart.Data("3", wb[3]),
                        new PieChart.Data("4", wb[4]),
                        new PieChart.Data("5", wb[5]));
        wbChart.setMaxSize(200, 200);
        wbChart.setData(wbData);
        wbChart.setClockwise(true);
        wbChart.setLabelsVisible(true);
        wbChart.setLegendVisible(true);
        wbChart.setLegendSide(Side.RIGHT);
        wbChart.setLayoutX(spiceChart.getLayoutX() + 600);
        wbChart.setLayoutY(spiceChart.getLayoutY());

        ObservableList<PieChart.Data> wsData =
                FXCollections.observableArrayList(
                        new PieChart.Data("0", ws[0]),
                        new PieChart.Data("1", ws[1]),
                        new PieChart.Data("2", ws[2]),
                        new PieChart.Data("3", ws[3]),
                        new PieChart.Data("4", ws[4]),
                        new PieChart.Data("5", ws[5]));
        wsChart.setMaxSize(200, 200);
        wsChart.setData(wsData);
        wsChart.setClockwise(true);
        wsChart.setLabelsVisible(true);
        wsChart.setLegendVisible(true);
        wsChart.setLegendSide(Side.RIGHT);
        wsChart.setLayoutX(charactersChart.getLayoutX() +
600);
        wsChart.setLayoutY(charactersChart.getLayoutY());
    }
}
```

IX. Manager.java

```
import java.sql.*;
import java.util.ArrayList;
public class Manager {
    private static Connection connection = null;
    private ArrayList authorsBooks = new
```

```
ArrayList<String[]>();
    private ArrayList readBooks = new ArrayList<String[]>();
    private ArrayList readingBooks = new
ArrayList<String[]>();
    private ArrayList tbrBooks = new ArrayList<String[]>();
    private ArrayList search = new ArrayList<String[]>();
    private static String nameOfUser;
    private int entry;
    private String[] bookDetails = new String[8];
    private float bookVolume;



    public static void main(String[] args) {

        String url = "jdbc:mariadb://localhost:3306/BookNook";
        String user = "root";
        String password = "******"; //Password removed for
privacy purposes

        try{
            connection = DriverManager.getConnection(url,
user, password);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static int checkUser(String username, String
password) {
        try {
            String query1 = "SELECT username, password, name
FROM Authors WHERE username=? AND password=?";
            String query2 = "SELECT username, password, name
FROM Readers WHERE username=? AND password=?";
            PreparedStatement stmt1 =
connection.prepareStatement(query1);
            PreparedStatement stmt2 =
connection.prepareStatement(query2);
            stmt1.setString(1, username);
            stmt1.setString(2, password);
            stmt2.setString(1, username);
            stmt2.setString(2, password);
            ResultSet result1 = stmt1.executeQuery();
            ResultSet result2 = stmt2.executeQuery();


            while(result1.next()){
                String testUsername =
result1.getString("username");
                String testPassword =
result1.getString("password");
                if((testUsername.equals(username)) &&
(testPassword.equals(password))){
                    nameOfUser = result1.getString("name");
                    return 1;
                }
            }

            while (result2.next()){
                String testUsername =
result2.getString("username");
                String testPassword =
result2.getString("password");
                if((testUsername.equals(username)) &&
(testPassword.equals(password))) {
                    nameOfUser = result2.getString("name");
                    return 2;
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return 0;
    }

    public void printAuthorsBooks(String username) {
        try {
            if(!authorsBooks.isEmpty()){
                authorsBooks.clear();
            }
            String query = "SELECT ISBN, title, author,
series, volume, publisher, genre, categories, synopsis FROM
(BookAuthor ba NATURAL JOIN BookDetails bd) WHERE
ba.authorUsername = ? AND ba.title = bd.title AND ba.author =
bd.author";
            PreparedStatement statement =
connection.prepareStatement(query);
            statement.setString(1, username);
            ResultSet result = statement.executeQuery();

            while(result.next()){
                String[] book = new String[5];
                book[0] = result.getString("title");
                book[1] = result.getString("series");
                book[2] =
String.valueOf(result.getFloat("volume"));
                book[3] = result.getString("ISBN");
                book[4] = result.getString("publisher");

                authorsBooks.add(book);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public String[] printProfile(String username, int user) {
        String[] profile = new String[3];
        if (user == 1) {
            try {
                String query = "SELECT * FROM Authors WHERE
username = ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                ResultSet results = stmt.executeQuery();
                while (results.next()) {
                    profile[0] = results.getString("name");
                    profile[1] =
results.getString("username");
                    profile[2] =
results.getString("password");
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        } else {
            try {
                String query = "SELECT * FROM Readers WHERE
username = ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                ResultSet results = stmt.executeQuery();
                while (results.next()) {
                    profile[0] = results.getString("name");
                    profile[1] =
results.getString("username");
                    profile[2] =
results.getString("password");
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }


        return profile;
    }

    public void createUser(String username, String password,
String name, int user) {
        try {
            if(user == 1) {
                String authorQuery = "INSERT into Authors
VALUES (?, ?, ?)";
                PreparedStatement authorStatement =
connection.prepareStatement(authorQuery);
                authorStatement.setString(1, username);
                authorStatement.setString(2, password);
                authorStatement.setString(3, name);
                int rows = authorStatement.executeUpdate();
                if(rows != 1) {
                    System.out.print("not updated");
                }
            } else if (user == 2) {
                String readerQuery = "INSERT into Readers
VALUES (?, ?, ?)";
                PreparedStatement readerStatement =
connection.prepareStatement(readerQuery);
                readerStatement.setString(1, username);
                readerStatement.setString(2, password);
                readerStatement.setString(3, name);
                int rows = readerStatement.executeUpdate();
                if(rows != 1) {
                    System.out.print("not updated");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public void createBook(String title, String author, String
series, Float volume, String ISBN, String genre, String
categories, String publisher, String synopsis, String
username) {
        try {
            String query = "SELECT * FROM BookAuthor WHERE
ISBN = ?";
            PreparedStatement statement =
connection.prepareStatement(query);
            statement.setString(1, ISBN);
            ResultSet results = statement.executeQuery();
            if(results.next()){
                String update = "UPDATE BookAuthor SET
authorUsername = ? WHERE ISBN = ?";
                PreparedStatement stmt =
connection.prepareStatement(update);
                stmt.setString(1, username);
                stmt.setString(2, ISBN);
                stmt.executeUpdate();
            } else {
                String baQuery = "INSERT into BookAuthor
VALUES (?, ?, ?, ?, ?)";
                PreparedStatement baStmt =
connection.prepareStatement(baQuery);
                baStmt.setString(1, ISBN);
                baStmt.setString(2, title);
                baStmt.setString(3, author);
                baStmt.setString(4, publisher);
                baStmt.setString(5, username);
                int baRow = baStmt.executeUpdate();
                if(baRow != 1) {
                    System.out.println("not updated");
                }

                String bdQuery = "INSERT into BookDetails 
VALUES (?, ?, ?, ?, ?, ?, ?)";
                PreparedStatement bdStmt = 
connection.prepareStatement(bdQuery);
                bdStmt.setString(1, title);
                bdStmt.setString(2, author);
                bdStmt.setString(3, series);
                bdStmt.setFloat(4, volume);
                bdStmt.setString(5, genre);
                bdStmt.setString(6, categories);
                bdStmt.setString(7, synopsis);
                int bdRow = bdStmt.executeUpdate();
                if(bdRow != 1) {
                    System.out.println("not updated");
                }
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

    }

    public void recallBook(String ISBN) {
        try {
            String query = "SELECT title, author, series, 
volume, ISBN, genre, categories, publisher, synopsis FROM 
(BookAuthor ba natural join BookDetails bd) WHERE ba.title = 
bd.title AND ba.author = bd.author AND ba.ISBN = ?";
            PreparedStatement stmt = 
connection.prepareStatement(query);
            stmt.setString(1, ISBN);
            ResultSet results = stmt.executeQuery();
            while(results.next()){
                bookDetails[0] =  results.getString("title");
                bookDetails[1] = results.getString("author");
                bookDetails[2] = results.getString("series");
                bookVolume = results.getFloat("volume");
                bookDetails[3] = results.getString("genre");
                bookDetails[4] = 
results.getString("categories");
                bookDetails[5] = 
results.getString("publisher");
                bookDetails[6] = 
results.getString("synopsis");
            }
        } catch (SQLException e) {
        }
    }

    public void updateBook(String title, String author, String
series, Float volume, String ISBN, String genre, String
categories, String publisher, String synopsis, String oldISBN,
String username) {
        try {
            String query = "UPDATE (BookAuthor ba NATURAL JOIN
BookDetails bd) SET title = ?, author = ?, series = ?, volume
= ?, ISBN = ?, genre = ?, categories = ?, publisher = ?,
synopsis = ? WHERE ISBN = ? AND ba.authorUsername = ?";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, title);
            stmt.setString(2, author);
            stmt.setString(3, series);
            stmt.setFloat(4, volume);
            stmt.setString(5, ISBN);
            stmt.setString(6, genre);
            stmt.setString(7, categories);
            stmt.setString(8, publisher);
            stmt.setString(9, synopsis);
            stmt.setString(10, oldISBN);
            stmt.setString(11, username);
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void deleteBook(String ISBN) {
        try {
            String query1 = "DELETE ba.*, bd.* FROM BookAuthor
ba, BookDetails bd WHERE ba.title = bd.title AND ba.author =
bd.author AND ba.isbn = ?";
            PreparedStatement stmt1 =
connection.prepareStatement(query1);
            stmt1.setString(1, ISBN);
            stmt1.executeUpdate();
            String query2 = "DELETE r.*, wr.* FROM Reviews r,
WritesReview wr WHERE r.reviewID = wr.reviewID and r.ISBN =
?";
            PreparedStatement stmt2 =
connection.prepareStatement(query2);
            stmt2.setString(1, ISBN);
            stmt2.executeUpdate();
            String query3 = "DELETE rb.* FROM ReadsBook rb
WHERE rb.ISBN = ?";
            PreparedStatement stmt3 =
connection.prepareStatement(query3);
            stmt3.setString(1, ISBN);
            stmt3.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void updateAuthor(String name, String username,
String password, String oldUsername) {
        try {
            String query1 = "UPDATE Authors SET name = ?,
username = ?, password = ? WHERE username = ?";
            PreparedStatement stmt1 =
connection.prepareStatement(query1);
            stmt1.setString(1, name);
            stmt1.setString(2, username);
            stmt1.setString(3, password);
            stmt1.setString(4, oldUsername);
            stmt1.executeUpdate();

            String query2 = "UPDATE BookAuthor SET
authorUsername = ? WHERE authorUsername = ?";
            PreparedStatement stmt2 =
connection.prepareStatement(query2);
            stmt2.setString(1, username);
            stmt2.setString(2, oldUsername);
            stmt2.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void updateReader(String name, String username,
String password, String oldUsername) {
        try {
            String query1 = "UPDATE Readers SET name = ?,
username = ?, password = ? WHERE username = ?";
            PreparedStatement stmt1 =
connection.prepareStatement(query1);
            stmt1.setString(1, name);
            stmt1.setString(2, username);
            stmt1.setString(3, password);
            stmt1.setString(4, oldUsername);
            stmt1.executeUpdate();

            String query2 = "UPDATE ReadsBook SET username = ?
WHERE username = ?";
            PreparedStatement stmt2 =
connection.prepareStatement(query2);
            stmt2.setString(1, username);
            stmt2.setString(2, oldUsername);
            stmt2.executeUpdate();

            String query3 = "UPDATE WritesReview SET username
```

```
= ? WHERE username = ?";
            PreparedStatement stmt3 =
connection.prepareStatement(query3);
            stmt3.setString(1, username);
            stmt3.setString(2, oldUsername);
            stmt3.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }

    }

    public void deleteProfile(String username, int user) {
        try {
            if (user == 1 ) {
                String query = "DELETE a.* FROM Authors a
WHERE username = ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                stmt.executeUpdate();

                String query2 = "UPDATE BookAuthor SET
authorUsername = ? WHERE authorUsername = ?";
                PreparedStatement stmt2 =
connection.prepareStatement(query2);
                stmt2.setString(1, null);
                stmt2.setString(2, username);
                stmt2.executeUpdate();
            } else {
                String query = "DELETE r.* FROM Readers r
WHERE username = ?";
                PreparedStatement stmt1 =
connection.prepareStatement(query);
                stmt1.setString(1, username);
                stmt1.executeUpdate();

                String query2 = "DELETE wr.*, r.* FROM
WritesReview wr, Reviews r WHERE wr.reviewID = r.reviewID AND
wr.username = ?";
                PreparedStatement stmt2 =
connection.prepareStatement(query2);
                stmt2.setString(1, username);
                stmt2.executeUpdate();

                String query3 = "DELETE rb.* FROM ReadsBook rb
WHERE username = ?";
                PreparedStatement stmt3 =
connection.prepareStatement(query3);
                stmt3.setString(1, username);
                stmt3.executeUpdate();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void pullReadBooks(String username) {
        try {
            String query = "SELECT ba.title, ba.author,
bd.series, bd.volume, ba.ISBN, ba.publisher FROM ReadsBook rb,
BookAuthor ba, BookDetails bd WHERE ba.title = bd.title AND
ba.author = bd.author AND ba.ISBN = rb.ISBN AND username = ?
and readingStatus = 'read'";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, username);
            ResultSet results = stmt.executeQuery();
            while (results.next()) {
                String[] book = new String[6];
                book[0] = results.getString("title");
                book[1] = results.getString("author");
                book[2] = results.getString("series");
                book[3] =
String.valueOf(results.getFloat("volume"));
                book[4] = results.getString("ISBN");
                book[5] = results.getString("publisher");
                readBooks.add(book);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void pullReadingBooks(String username) {
        try {
            String query = "SELECT ba.title, ba.author,
bd.series, bd.volume, ba.ISBN, ba.publisher FROM ReadsBook rb,
BookAuthor ba, BookDetails bd WHERE ba.title = bd.title AND
ba.author = bd.author AND ba.ISBN = rb.ISBN AND username = ?
and readingStatus = 'reading'";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, username);
            ResultSet results = stmt.executeQuery();
            while (results.next()) {
                String[] book = new String[6];
                book[0] = results.getString("title");
                book[1] = results.getString("author");
                book[2] = results.getString("series");
                book[3] =
String.valueOf(results.getFloat("volume"));
                book[4] = results.getString("ISBN");
                book[5] = results.getString("publisher");
                readingBooks.add(book);
            }
        } catch (SQLException e) {
             e.printStackTrace();
        }
    }

    public void pullTBR(String username) {
        try {
            String query = "SELECT ba.title, ba.author,
bd.series, bd.volume, ba.ISBN, ba.publisher FROM ReadsBook rb,
BookAuthor ba, BookDetails bd WHERE ba.title = bd.title AND
ba.author = bd.author AND ba.ISBN = rb.ISBN AND username = ?
and readingStatus = 'tbr'";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, username);
            ResultSet results = stmt.executeQuery();
            while (results.next()) {
                String[] book = new String[6];
                book[0] = results.getString("title");
                book[1] = results.getString("author");
                book[2] = results.getString("series");
                book[3] =
String.valueOf(results.getFloat("volume"));
                book[4] = results.getString("ISBN");
                book[5] = results.getString("publisher");
                tbrBooks.add(book);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void pullSearch(String newSearch) {
        try {
            String searchString = "%" + newSearch + "%";
            String query = "SELECT ba.title, ba.author,
bd.series, bd.volume, ba.ISBN, ba.publisher, bd.synopsis FROM
BookAuthor ba, BookDetails bd WHERE ba.title = bd.title AND
ba.author = bd.author AND ba.title LIKE ?";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, searchString);
            ResultSet results = stmt.executeQuery();
            while(results.next()) {
                String[] searchResult = new String[7];
                searchResult[0] =
results.getString("ba.title");
                searchResult[1] =
results.getString("ba.author");
                searchResult[2] =
results.getString("bd.series");
                searchResult[3] =
String.valueOf(results.getString("bd.volume"));
```

```
                searchResult[4] =
results.getString("ba.ISBN");
                searchResult[5] =
results.getString("ba.publisher");
                searchResult[6] =
results.getString("bd.synopsis");
                search.add(searchResult);
            }

            String query2 = "SELECT ba.title, ba.author,
bd.series, bd.volume, ba.ISBN, ba.publisher, bd.synopsis FROM
BookAuthor ba, BookDetails bd WHERE ba.title = bd.title AND
ba.author = bd.author AND ba.author LIKE ?";
            PreparedStatement stmt2 =
connection.prepareStatement(query2);
            stmt2.setString(1, searchString);
            ResultSet results2 = stmt2.executeQuery();
            while(results2.next()) {
                String[] searchResult = new String[7];
                searchResult[0] =
results2.getString("ba.title");
                searchResult[1] =
results2.getString("ba.author");
                searchResult[2] =
results2.getString("bd.series");
                searchResult[3] =
String.valueOf(results2.getString("bd.volume"));
                searchResult[4] =
results2.getString("ba.ISBN");
                searchResult[5] =
results2.getString("ba.publisher");
                searchResult[6] =
results2.getString("bd.synopsis");
                search.add(searchResult);
            }

            String query3 = "SELECT ba.title, ba.author,
bd.series, bd.volume, ba.ISBN, ba.publisher, bd.synopsis FROM
BookAuthor ba, BookDetails bd WHERE ba.title = bd.title AND
ba.author = bd.author AND bd.series LIKE ?";
            PreparedStatement stmt3 =
connection.prepareStatement(query3);
            stmt3.setString(1, searchString);
            ResultSet results3 = stmt3.executeQuery();
            while(results3.next()) {
                String[] searchResult = new String[7];
                searchResult[0] =
results3.getString("ba.title");
                searchResult[1] =
results3.getString("ba.author");
                searchResult[2] =
results3.getString("bd.series");
                searchResult[3] =
String.valueOf(results3.getString("bd.volume"));
                searchResult[4] =
results3.getString("ba.ISBN");
                searchResult[5] =
results3.getString("ba.publisher");
                searchResult[6] =
results3.getString("bd.synopsis");
                search.add(searchResult);
            }


        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public String getReadingStatus(String username, String
ISBN) {
        String status = null;
        entry = 0;
        try {
            String query = "SELECT readingStatus FROM
ReadsBook rb WHERE username = ? AND ISBN = ?";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, username);
            stmt.setString(2, ISBN);
            ResultSet results = stmt.executeQuery();
            while(results.next()) {
                String result =
results.getString("readingStatus");

                if(result.equalsIgnoreCase("reading")) {
                    status = "Reading";
                } else if(result.equalsIgnoreCase("read")) {
                    status = "Read";
                } else if(result.equalsIgnoreCase("tbr")) {
                    status = "TBR";
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return status;
    }

    public void setReadingStatus(String username, String ISBN,
String readingStatus) {
        try {
            String query = "INSERT INTO ReadsBook VALUES (?,
?, ?)";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, username);
            stmt.setString(2, ISBN);
            stmt.setString(3, readingStatus);
            int row = stmt.executeUpdate();
            if(row != 1) {
                System.out.println("not updated");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void removeReadingStatus(String username, String
ISBN) {
        try {
            String query = "DELETE rb.* FROM ReadsBook rb
WHERE username = ? AND ISBN = ?";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, username);
            stmt.setString(2, ISBN);
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void updateReadingStatus(String username, String
ISBN, String readingStatus) {
        try {
            String query = "UPDATE ReadsBook SET readingStatus
= ? WHERE username = ? AND ISBN = ?";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, readingStatus);
            stmt.setString(2, username);
            stmt.setString(3, ISBN);
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void createReview(String username, String ISBN,
Float storyline, Float plot, Float setting, Float spice, Float
characters, Float wb, Float ws, String dateStarted, String
dateFinished, String thoughts) {
        int oldReviewID = 0;
        try {
            String query = "SELECT MAX(reviewID) FROM
Reviews";
            Statement stmt =
connection.prepareStatement(query);
            ResultSet result = stmt.executeQuery(query);
            while(result.next()) {
                oldReviewID = result.getInt("MAX(reviewID)");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        int newReviewID = oldReviewID + 1;

        try {
            java.sql.Date ds =
java.sql.Date.valueOf(dateStarted);
            java.sql.Date df =
java.sql.Date.valueOf(dateFinished);
            String query1 = "INSERT INTO Reviews VALUES (?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement stmt1 =
connection.prepareStatement(query1);
            stmt1.setInt(1, newReviewID);
            stmt1.setString(2, ISBN);
            stmt1.setFloat(3, storyline);
            stmt1.setFloat(4, plot);
            stmt1.setFloat(5, setting);
            stmt1.setFloat(6, spice);
            stmt1.setFloat(7, characters);
            stmt1.setFloat(8, wb);
            stmt1.setFloat(9, ws);
            stmt1.setDate(10, ds);
            stmt1.setDate(11, df);
            stmt1.setString(12, thoughts);
            int row = stmt1.executeUpdate();
            if(row != 1) {
                System.out.print("not updated");
            }
        } catch (SQLException e){
            e.printStackTrace();
        }

        try {
            String query2 = "INSERT INTO WritesReview VALUES
(?, ?)";
            PreparedStatement stmt2 =
connection.prepareStatement(query2);
            stmt2.setString(1, username);
            stmt2.setInt(2, newReviewID);
            int row2 = stmt2.executeUpdate();
            if(row2 != 1) {
                System.out.print("not updated");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public String[] pullReviewDetails(String username, String
ISBN) {
        String[] review = new String[10];
        try {
            String query = "SELECT r.storylineRating,
r.plotRating, r.settingRating, r.spiceRating,
r.charactersRating, r.wbRating, r.wsRating, r.dateStarted,
r.dateFinished, r.writtenReview from Reviews r, WritesReview
wr WHERE r.reviewID = wr.reviewID AND wr.username = ? AND
r.ISBN = ?";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, username);
            stmt.setString(2, ISBN);
            ResultSet results = stmt.executeQuery();
            while(results.next()) {
                review[0] =
String.valueOf(results.getFloat("storylineRating"));
                review[1] =
String.valueOf(results.getFloat("plotRating"));
                review[2] =
String.valueOf(results.getFloat("settingRating"));
                review[3] =
String.valueOf(results.getFloat("spiceRating"));
                review[4] =
String.valueOf(results.getFloat("charactersRating"));
                review[5] =
String.valueOf(results.getFloat("wbRating"));
                review[6] =
String.valueOf(results.getFloat("wsRating"));
                review[7] =
String.valueOf(results.getDate("dateStarted"));
                review[8] =
String.valueOf(results.getDate("dateFinished"));
                review[9] =
results.getString("writtenReview");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return review;
    }

    public void updateReview(String username, String ISBN,
Float storyline, Float plot, Float setting, Float spice, Float
characters, Float wb, Float ws, String dateStarted, String
dateFinished, String thoughts) {
        int reviewID = 0;
        try {
            String find = "SELECT r.reviewID FROM Reviews r,
WritesReview wr WHERE r.reviewID = wr.reviewID AND r.ISBN = ?
AND wr.username = ?";
            PreparedStatement findstmt =
connection.prepareStatement(find);
            findstmt.setString(1, ISBN);
            findstmt.setString(2, username);
            ResultSet result = findstmt.executeQuery();
            while(result.next()) {
                reviewID = result.getInt("reviewID");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        try {
            String query = "UPDATE Reviews SET storylineRating
= ?, plotRating = ?, settingRating = ?, spiceRating = ?,
charactersRating = ?, wbRating = ?, wsRating = ?, dateStarted
= ?, dateFinished = ?, writtenReview = ? WHERE reviewID = ?";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setFloat(1, storyline);
            stmt.setFloat(2, plot);
            stmt.setFloat(3, setting);
            stmt.setFloat(4, spice);
            stmt.setFloat(5, characters);
            stmt.setFloat(6, wb);
            stmt.setFloat(7, ws);
            java.sql.Date ds =
java.sql.Date.valueOf(dateStarted);
            stmt.setDate(8, ds);
            java.sql.Date df =
java.sql.Date.valueOf(dateFinished);
            stmt.setDate(9, df);
            stmt.setString(10, thoughts);
            stmt.setInt(11, reviewID);
            stmt.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void deleteReview(String username, String ISBN) {
        try {
            String query = "DELETE r.*, wr.* FROM Reviews r,
WritesReview wr WHERE r.reviewID = wr.reviewID AND r.ISBN = ?
AND wr.username = ?";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, ISBN);
            stmt.setString(2, username);
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public int verify(int user, String username) {
        int duplicate = 0;
        try {
            if (user == 1) {
                String query = "SELECT username FROM Authors
WHERE username = ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                ResultSet result = stmt.executeQuery();
                while (result.next()) {
                    duplicate = 1;
                }
            } else {
                String query = "SELECT username FROM Readers
WHERE username = ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                ResultSet result = stmt.executeQuery();
                while (result.next()) {
                    duplicate = 1;
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return duplicate;
    }

    public ArrayList<int[]> getAuthorReviews(String ISBN) {
        ArrayList reviews = new ArrayList<int[]>();
        int[] storylineReviews = new int[6];
        int[] plotReviews = new int[6];
        int[] settingReviews = new int[6];
        int[] spiceReviews = new int[6];
        int[] characterReviews = new int[6];
        int[] wbReviews = new int[6];
        int[] wsReviews = new int[6];

        try {
            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(reviewID) AS
storyline FROM Reviews WHERE ISBN = ? AND storylineRating >= ?
AND storylineRating < ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, ISBN);
                stmt.setInt(2, i);
                stmt.setInt(3, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("storyline");
                    storylineReviews[i] = count;
                }
            }
            reviews.add(storylineReviews);

            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(reviewID) AS plot
FROM Reviews WHERE ISBN = ? AND plotRating >= ? AND plotRating
< ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, ISBN);
                stmt.setInt(2, i);
                stmt.setInt(3, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("plot");
                    plotReviews[i] = count;
                }
            }
            reviews.add(plotReviews);

            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(reviewID) AS
setting FROM Reviews WHERE ISBN = ? AND settingRating >= ? AND
settingRating < ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, ISBN);
                stmt.setInt(2, i);
                stmt.setInt(3, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("setting");
                    settingReviews[i] = count;
                }
            }
            reviews.add(settingReviews);

            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(reviewID) AS
spice FROM Reviews WHERE ISBN = ? AND spiceRating >= ? AND
spiceRating < ?";
```

```
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, ISBN);
                stmt.setInt(2, i);
                stmt.setInt(3, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("spice");
                    spiceReviews[i] = count;
                }
            }
            reviews.add(spiceReviews);

            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(reviewID) AS
characters FROM Reviews WHERE ISBN = ? AND charactersRating >=
? AND charactersRating < ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, ISBN);
                stmt.setInt(2, i);
                stmt.setInt(3, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("characters");
                    characterReviews[i] = count;
                }
            }
            reviews.add(characterReviews);

            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(reviewID) AS wb
FROM Reviews WHERE ISBN = ? AND wbRating >= ? AND wbRating <
?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, ISBN);
                stmt.setInt(2, i);
                stmt.setInt(3, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("wb");
                    wbReviews[i] = count;
                }
            }
            reviews.add(wbReviews);

            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(reviewID) AS ws
FROM Reviews WHERE ISBN = ? AND wsRating >= ? AND wsRating <
```

```
?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, ISBN);
                stmt.setInt(2, i);
                stmt.setInt(3, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("ws");
                    wsReviews[i] = count;
                }
            }
            reviews.add(wsReviews);

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return reviews;
    }

    public ArrayList<int[]> getReaderReviews(String username,
String dateStarted, String dateFinished) {
        ArrayList reviews = new ArrayList<int[]>();
        int[] storylineReviews = new int[6];
        int[] plotReviews = new int[6];
        int[] settingReviews = new int[6];
        int[] spiceReviews = new int[6];
        int[] characterReviews = new int[6];
        int[] wbReviews = new int[6];
        int[] wsReviews = new int[6];
        int[] genres = new int[6];
        java.sql.Date ds = java.sql.Date.valueOf(dateStarted);
        java.sql.Date df =
java.sql.Date.valueOf(dateFinished);

        try {
            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(r.reviewID) AS
storyline FROM Reviews r, WritesReview wr WHERE r.reviewID =
wr.reviewID AND wr.username = ? AND r.dateStarted >= ? AND
r.dateFinished <= ? AND storylineRating >= ? AND
storylineRating < ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                stmt.setDate(2, ds);
                stmt.setDate(3, df);
                stmt.setInt(4, i);
                stmt.setInt(5, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("storyline");
```

```
                        storylineReviews[i] = count;
                    }
                }
                reviews.add(storylineReviews);

                for (int i = 0; i < 6; i++) {
                    int k = i + 1;
                    String query = "SELECT COUNT(r.reviewID) AS
plot FROM Reviews r, WritesReview wr WHERE r.reviewID =
wr.reviewID AND wr.username = ? AND r.dateStarted >= ? AND
r.dateFinished <= ? AND plotRating >= ? AND plotRating < ?";
                    PreparedStatement stmt =
connection.prepareStatement(query);
                    stmt.setString(1, username);
                    stmt.setDate(2, ds);
                    stmt.setDate(3, df);
                    stmt.setInt(4, i);
                    stmt.setInt(5, k);
                    ResultSet result = stmt.executeQuery();
                    while(result.next()){
                        int count = result.getInt("plot");
                        plotReviews[i] = count;
                    }
                }
                reviews.add(plotReviews);

                for (int i = 0; i < 6; i++) {
                    int k = i + 1;
                    String query = "SELECT COUNT(r.reviewID) AS
setting FROM Reviews r, WritesReview wr WHERE r.reviewID =
wr.reviewID AND wr.username = ? AND r.dateStarted >= ? AND
r.dateFinished <= ? AND settingRating >= ? AND settingRating <
?";
                    PreparedStatement stmt =
connection.prepareStatement(query);
                    stmt.setString(1, username);
                    stmt.setDate(2, ds);
                    stmt.setDate(3, df);
                    stmt.setInt(4, i);
                    stmt.setInt(5, k);
                    ResultSet result = stmt.executeQuery();
                    while(result.next()){
                        int count = result.getInt("setting");
                        settingReviews[i] = count;
                    }
                }
                reviews.add(settingReviews);

                for (int i = 0; i < 6; i++) {
                    int k = i + 1;
                    String query = "SELECT COUNT(r.reviewID) AS
spice FROM Reviews r, WritesReview wr WHERE r.reviewID =
wr.reviewID AND wr.username = ? AND r.dateStarted >= ? AND
```

```
r.dateFinished <= ? AND spiceRating >= ? AND spiceRating < ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                stmt.setDate(2, ds);
                stmt.setDate(3, df);
                stmt.setInt(4, i);
                stmt.setInt(5, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("spice");
                    spiceReviews[i] = count;
                }
            }
            reviews.add(spiceReviews);

            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(r.reviewID) AS
characters FROM Reviews r, WritesReview wr WHERE r.reviewID =
wr.reviewID AND wr.username = ? AND r.dateStarted >= ? AND
r.dateFinished <= ? AND charactersRating >= ? AND
charactersRating < ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                stmt.setDate(2, ds);
                stmt.setDate(3, df);
                stmt.setInt(4, i);
                stmt.setInt(5, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("characters");
                    characterReviews[i] = count;
                }
            }
            reviews.add(characterReviews);

            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(r.reviewID) AS wb
FROM Reviews r, WritesReview wr WHERE r.reviewID = wr.reviewID
AND wr.username = ? AND r.dateStarted >= ? AND r.dateFinished
<= ? AND wbRating >= ? AND wbRating < ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                stmt.setDate(2, ds);
                stmt.setDate(3, df);
                stmt.setInt(4, i);
                stmt.setInt(5, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("wb");
                    wbReviews[i] = count;
                }
            }
            reviews.add(wbReviews);

            for (int i = 0; i < 6; i++) {
                int k = i + 1;
                String query = "SELECT COUNT(r.reviewID) AS ws
FROM Reviews r, WritesReview wr WHERE r.reviewID = wr.reviewID
AND wr.username = ? AND r.dateStarted >= ? AND r.dateFinished
<= ? AND wsRating >= ? AND wsRating < ?";
                PreparedStatement stmt =
connection.prepareStatement(query);
                stmt.setString(1, username);
                stmt.setDate(2, ds);
                stmt.setDate(3, df);
                stmt.setInt(4, i);
                stmt.setInt(5, k);
                ResultSet result = stmt.executeQuery();
                while(result.next()){
                    int count = result.getInt("ws");
                    wsReviews[i] = count;
                }
            }
            reviews.add(wsReviews);

            String query1 = "SELECT COUNT(r.reviewID) AS
romance FROM Reviews r, WritesReview wr, BookDetails bd,
BookAuthor ba WHERE r.reviewID = wr.reviewID AND r.ISBN =
ba.ISBN AND ba.title = bd.title AND ba.author = bd.author AND
wr.username = ? AND r.dateStarted >= ? AND r.dateFinished <= ?
AND bd.genre = 'romance'";
            PreparedStatement stmt1 =
connection.prepareStatement(query1);
            stmt1.setString(1, username);
            stmt1.setDate(2, ds);
            stmt1.setDate(3, df);
            ResultSet result1 = stmt1.executeQuery();
            while (result1.next()) {
                genres[0] = result1.getInt("romance");
            }

            String query2 = "SELECT COUNT(r.reviewID) AS
fantasy FROM Reviews r, WritesReview wr, BookDetails bd,
BookAuthor ba WHERE r.reviewID = wr.reviewID AND r.ISBN =
ba.ISBN AND ba.title = bd.title AND ba.author = bd.author AND
wr.username = ? AND r.dateStarted >= ? AND r.dateFinished <= ?
AND bd.genre = 'fantasy'";
            PreparedStatement stmt2 =
connection.prepareStatement(query2);
            stmt2.setString(1, username);
            stmt2.setDate(2, ds);
            stmt2.setDate(3, df);
            ResultSet result2 = stmt2.executeQuery();
            while (result2.next()) {
                genres[1] = result2.getInt("fantasy");
            }

            String query3 = "SELECT COUNT(r.reviewID) AS
mystery FROM Reviews r, WritesReview wr, BookDetails bd,
BookAuthor ba WHERE r.reviewID = wr.reviewID AND r.ISBN =
ba.ISBN AND ba.title = bd.title AND ba.author = bd.author AND
wr.username = ? AND r.dateStarted >= ? AND r.dateFinished <= ?
AND bd.genre = 'mystery'";
            PreparedStatement stmt3 =
connection.prepareStatement(query3);
            stmt3.setString(1, username);
            stmt3.setDate(2, ds);
            stmt3.setDate(3, df);
            ResultSet result3 = stmt3.executeQuery();
            while (result3.next()) {
                genres[2] = result3.getInt("mystery");
            }

            String query4 = "SELECT COUNT(r.reviewID) AS
thriller FROM Reviews r, WritesReview wr, BookDetails bd,
BookAuthor ba WHERE r.reviewID = wr.reviewID AND r.ISBN =
ba.ISBN AND ba.title = bd.title AND ba.author = bd.author AND
wr.username = ? AND r.dateStarted >= ? AND r.dateFinished <= ?
AND bd.genre = 'thriller'";
            PreparedStatement stmt4 =
connection.prepareStatement(query4);
            stmt4.setString(1, username);
            stmt4.setDate(2, ds);
            stmt4.setDate(3, df);
            ResultSet result4 = stmt4.executeQuery();
            while (result4.next()) {
                genres[3] = result4.getInt("thriller");
            }

            String query5 = "SELECT COUNT(r.reviewID) AS
historical FROM Reviews r, WritesReview wr, BookDetails bd,
BookAuthor ba WHERE r.reviewID = wr.reviewID AND r.ISBN =
ba.ISBN AND ba.title = bd.title AND ba.author = bd.author AND
wr.username = ? AND r.dateStarted >= ? AND r.dateFinished <= ?
AND bd.genre = 'historical fiction'";
            PreparedStatement stmt5 =
connection.prepareStatement(query5);
            stmt5.setString(1, username);
            stmt5.setDate(2, ds);
            stmt5.setDate(3, df);
            ResultSet result5 = stmt5.executeQuery();
            while (result5.next()) {
                genres[4] = result5.getInt("historical");
            }
            String query6 = "SELECT COUNT(r.reviewID) AS
science FROM Reviews r, WritesReview wr, BookDetails bd,
BookAuthor ba WHERE r.reviewID = wr.reviewID AND r.ISBN =
ba.ISBN AND ba.title = bd.title AND ba.author = bd.author AND
wr.username = ? AND r.dateStarted >= ? AND r.dateFinished <= ?
AND bd.genre = 'science fiction'";
            PreparedStatement stmt6 =
connection.prepareStatement(query6);
            stmt6.setString(1, username);
            stmt6.setDate(2, ds);
            stmt6.setDate(3, df);
            ResultSet result6 = stmt6.executeQuery();
            while (result6.next()) {
                genres[5] = result6.getInt("science");
            }

            reviews.add(genres);

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return reviews;
    }

    public int getNumberBooks(String username, String
dateStart, String dateFinish) {
        int numberBooks = 0;
        java.sql.Date ds = java.sql.Date.valueOf(dateStart);
        java.sql.Date df = java.sql.Date.valueOf(dateFinish);

        try {
            String query = "SELECT COUNT(r.reviewID) AS count
FROM Reviews r, WritesReview wr WHERE r.reviewID = wr.reviewID
AND wr.username = ? AND r.dateStarted >= ? AND r.dateFinished
<= ?";
            PreparedStatement stmt =
connection.prepareStatement(query);
            stmt.setString(1, username);
            stmt.setDate(2, ds);
            stmt.setDate(3, df);
            ResultSet result = stmt.executeQuery();
            while(result.next()) {
                numberBooks = result.getInt("count");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return numberBooks;
    }

    public ArrayList<String[]> getAuthorsBooks() {
        return authorsBooks;
    }

    public ArrayList<String[]> getReadBooks() {
        return readBooks;
    }

    public ArrayList<String[]> getReadingBooks() {
        return readingBooks;
    }

    public ArrayList<String[]> getTBRBooks() {
        return tbrBooks;
    }

    public ArrayList<String[]> getSearch() {
        return search;
    }

    public String getNameOfUser() {
        return nameOfUser;
    }

    public void setNameOfUser(String nameOfUser) {
        this.nameOfUser = nameOfUser;
    }

    public String[] getBookDetails() {
        return bookDetails;
    }

    public Float getBookVolume() {
        return bookVolume;
    }
}
```

X. ReaderHomeScreen.java

```
import javafx.scene.control.Menu;
import javafx.scene.control.MenuBar;
import javafx.scene.control.MenuItem;
import javafx.scene.layout.BorderPane;

public class ReaderHomeScreen extends BorderPane {
    private MenuBar menu;
    private MenuItem menuBook,menuRead, menuReading, menuTBR;
    private MenuItem menuSeeReviews, menuUpdateReview,
menuDeleteReview;
    private MenuItem menuUpdateProfile, menuDeleteProfile;
    private MenuItem menuLogout;
    public ReaderHomeScreen() {
        createReaderHomeScreen();
    }
    public void createReaderHomeScreen() {
        this.setStyle("-fx-background-color: antiquewhite;");
        menu = new MenuBar();
        Menu menuBooks = new Menu("Books");
        menuBook = new MenuItem("Search Books");
        menuRead = new MenuItem("Read Books");
        menuReading = new MenuItem("Currently Reading");
        menuTBR = new MenuItem("TBR");
        menuBooks.getItems().addAll(menuBook, menuRead,
menuReading, menuTBR);
        Menu menuReviews =  new Menu("Reviews");
        menuUpdateReview = new MenuItem("Update Review");
        menuDeleteReview = new MenuItem("Delete Review");
        menuSeeReviews = new MenuItem("See Reviews");
        menuReviews.getItems().addAll(menuUpdateReview,
menuDeleteReview, menuSeeReviews);
        Menu menuSettings = new Menu("Settings");
        menuUpdateProfile = new MenuItem("Update Profile");
        menuDeleteProfile = new MenuItem("Delete Profile");
        menuSettings.getItems().addAll(menuUpdateProfile,
menuDeleteProfile);
        Menu logout = new Menu("Logout");
        menuLogout = new MenuItem("Logout");
        logout.getItems().addAll(menuLogout);
        menu.getMenus().addAll(menuBooks, menuReviews,
menuSettings, logout);
        this.setTop(menu);
    }
    public MenuItem getMenuBook() {
        return menuBook;
    }
    public MenuItem getMenuRead() {
        return menuRead;
    }
    public MenuItem getMenuReading() {
        return menuReading;
    }
    public MenuItem getMenuTBR() { return menuTBR; }
    public MenuItem getMenuUpdateReview() {
        return menuUpdateReview;
    }
    public MenuItem getMenuDeleteReview() {
        return menuDeleteReview;
    }
    public MenuItem getMenuUpdateProfile() {
        return menuUpdateProfile;
    }
    public MenuItem getMenuDeleteProfile() {
        return menuDeleteProfile;
    }
    public MenuItem getMenuSeeReviews() {
        return menuSeeReviews;
    }
    public MenuItem getMenuLogout() {
        return menuLogout;
    }
    public MenuBar getMenu() {
        return menu;
    }
}
```

XI. ReaderProfileUpdate.java

```
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.Pane;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
public class ReaderProfileUpdate extends Pane {

    private TextField name = new TextField();
    private TextField username = new TextField();
    private TextField password = new TextField();
    private Button update = new Button("Update Profile");

    public ReaderProfileUpdate() {
        createReaderProfileUpdate();
    }

    public void createReaderProfileUpdate() {
        Label edit = new Label("Please edit any or all
fields!");
        edit.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));

edit.layoutXProperty().bind(this.widthProperty().subtract(edit
.widthProperty()).divide(2));
        edit.setLayoutY(100);

        this.setStyle("-fx-background-color: antiquewhite;");

name.layoutXProperty().bind(this.widthProperty().subtract(name
.widthProperty()).divide(2));
        name.setLayoutY(200.0);

        Label nameLabel = new Label("Preferred Name");

nameLabel.layoutXProperty().bind(this.widthProperty().subtract
(name.widthProperty()).divide(2));
        nameLabel.setLayoutY(name.getLayoutY() - 20.0);


username.layoutXProperty().bind(this.widthProperty().subtract(
username.widthProperty()).divide(2));
        username.setLayoutY(name.getLayoutY() + 75.0);

        Label usernameLabel = new Label("Username");

usernameLabel.layoutXProperty().bind(this.widthProperty().subt
ract(username.widthProperty()).divide(2));
        usernameLabel.setLayoutY(username.getLayoutY() -
20.0);
password.layoutXProperty().bind(this.widthProperty().subtract(
password.widthProperty()).divide(2));
        password.setLayoutY(username.getLayoutY() + 75.0);

        Label passwordLabel = new Label("Password");

passwordLabel.layoutXProperty().bind(this.widthProperty().subt
ract(password.widthProperty()).divide(2));
        passwordLabel.setLayoutY(password.getLayoutY() -
20.0);


update.layoutXProperty().bind(this.widthProperty().subtract(pa
ssword.widthProperty()).divide(2));
        update.setLayoutY(password.getLayoutY() + 75.0);

        this.getChildren().addAll(name, nameLabel, username,
usernameLabel, password, passwordLabel, edit, update);

    }
    public TextField getUsername() {
        return username;
    }
    public TextField getName() {
        return name;
    }
    public TextField getPassword() {
        return password;
    }
    public Button getUpdate() {
        return update;
    }
}
```

XII. ReaderReviewPane.java

```
import javafx.geometry.Pos;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;

public class ReaderReviewPane extends BorderPane {
    Pane pane = new Pane();
    private TextField dsMonth = new TextField();
    private TextField dsDay = new TextField();
    private TextField dsYear = new TextField();
    private TextField dfMonth = new TextField();
    private TextField dfDay = new TextField();
    private TextField dfYear = new TextField();
    private Button getReviews = new Button("Get Review
Breakdown");

    public ReaderReviewPane() {
        createReaderReviewPane();
    }
    public void createReaderReviewPane() {
        pane.setStyle("-fx-background-color: antiquewhite;");
        Label enter = new Label("Please enter dates to see the
review breakdown:");
        enter.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 14));

enter.layoutXProperty().bind(pane.widthProperty().subtract(ent
er.widthProperty()).divide(2));
        enter.setLayoutY(25);
        pane.getChildren().add(enter);

        Label dateStartedText = new Label("Start: ");
        dateStartedText.setLayoutY(enter.getLayoutY() + 50);
        dateStartedText.setLayoutX(510);
        dsMonth.setLayoutX(dateStartedText.getLayoutX() + 50);
        dsMonth.setLayoutY(dateStartedText.getLayoutY() - 3);
        dsMonth.setPrefWidth(30);
        dsMonth.setAlignment(Pos.CENTER);
        Label date1 = new Label("/ ");
        date1.setLayoutX(dsMonth.getLayoutX() + 45);
        date1.setLayoutY(dateStartedText.getLayoutY());
        dsDay.setLayoutX(dsMonth.getLayoutX() + 60);
        dsDay.setLayoutY(dateStartedText.getLayoutY() - 3);
        dsDay.setPrefWidth(30);
        dsDay.setAlignment(Pos.CENTER);
        Label date2 = new Label("/ ");
        date2.setLayoutX(dsDay.getLayoutX() + 45);
        date2.setLayoutY(dateStartedText.getLayoutY());
        dsYear.setLayoutX(dsDay.getLayoutX() + 60);
        dsYear.setLayoutY(dateStartedText.getLayoutY() - 3);
        dsYear.setPrefWidth(50);
        dsYear.setAlignment(Pos.CENTER);
        pane.getChildren().addAll(dateStartedText, dsMonth,
date1, dsDay, date2, dsYear);

        Label dateFinishedText = new Label("End: ");

dateFinishedText.setLayoutY(dateStartedText.getLayoutY());
        dateFinishedText.setLayoutX(dsYear.getLayoutX() +
100);
        dfMonth.setLayoutX(dateFinishedText.getLayoutX() +
50);
        dfMonth.setLayoutY(dateFinishedText.getLayoutY() - 3);
        dfMonth.setPrefWidth(30);
        dfMonth.setAlignment(Pos.CENTER);
        Label date3 = new Label("/ ");
        date3.setLayoutX(dfMonth.getLayoutX() + 45);
        date3.setLayoutY(dateStartedText.getLayoutY());
        dfDay.setLayoutX(dfMonth.getLayoutX() + 60);
        dfDay.setLayoutY(dateFinishedText.getLayoutY() - 3);
        dfDay.setPrefWidth(30);
        dfDay.setAlignment(Pos.CENTER);
        Label date4 = new Label("/ ");
        date4.setLayoutX(dfDay.getLayoutX() + 45);
        date4.setLayoutY(dateFinishedText.getLayoutY());
        dfYear.setLayoutX(dfDay.getLayoutX() + 60);
        dfYear.setLayoutY(dateFinishedText.getLayoutY() - 3);
        dfYear.setPrefWidth(50);
        dfYear.setAlignment(Pos.CENTER);
        pane.getChildren().addAll(dateFinishedText, dfMonth,
date3, dfDay, date4, dfYear);


getReviews.layoutXProperty().bind(pane.widthProperty().subtrac
t(getReviews.widthProperty()).divide(2));
        getReviews.setLayoutY(dfYear.getLayoutY() + 75);
        pane.getChildren().add(getReviews);
        this.setTop(pane);
    }

    public TextField getDsMonth() {
        return dsMonth;
    }

    public TextField getDsDay() {
        return dsDay;
    }

    public TextField getDsYear() {
        return dsYear;
    }

    public TextField getDfMonth() {
        return dfMonth;
    }

    public TextField getDfDay() {
        return dfDay;
    }

    public TextField getDfYear() {
        return dfYear;
    }

    public Button getGetReviews() {
        return getReviews;
    }

    public Pane getPane() {
        return pane;
    }
}
```

XIII. ReadingStatusPane.java

```
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.Pane;

public class ReadingStatusPane extends Pane {
    private Button search = new Button("Search");
    private TextField ISBN = new TextField();
    private  ComboBox readingStatus = new ComboBox();
    private Button setReadingStatus = new Button("Set Reading
Status");
    private Button editReadingStatus = new Button("Edit
Reading Status");
    private Label setReadingStatusText = new Label("Please
select/edit your reading status:");
    private Button removeStatus = new Button("Delete Status");

    public ReadingStatusPane() {
        createReadingStatusPane();
    }

    public void createReadingStatusPane() {
        this.setStyle("-fx-background-color: antiquewhite;");
        Label text = new Label("Please enter the ISBN of the
book to set/change reading status");

text.layoutXProperty().bind(this.widthProperty().subtract(text
.widthProperty()).divide(2));
        text.setLayoutY(100);

ISBN.layoutXProperty().bind(this.widthProperty().subtract(ISBN
.widthProperty()).divide(2));
        ISBN.setLayoutY(text.getLayoutY() + 50);

search.layoutXProperty().bind(this.widthProperty().subtract(se
arch.widthProperty()).divide(2));
        search.setLayoutY(ISBN.getLayoutY() + 100);

setReadingStatusText.layoutXProperty().bind(this.widthProperty
().subtract(setReadingStatusText.widthProperty()).divide(2));
        setReadingStatusText.setLayoutY(ISBN.getLayoutY() +
50);
        this.getChildren().addAll(text, ISBN, search,
setReadingStatusText);
        setReadingStatusText.setVisible(false);
        readingStatus.getItems().addAll(
                "Read",
                "Reading",
                "TBR"
        );
        readingStatus.setPrefWidth(250);

readingStatus.layoutXProperty().bind(this.widthProperty().subt
ract(readingStatus.widthProperty()).divide(2));
        readingStatus.setLayoutY(this.getISBN().getLayoutY() +
100);

setReadingStatus.layoutXProperty().bind(this.widthProperty().s
ubtract(setReadingStatus.widthProperty()).divide(2.5));
        setReadingStatus.setLayoutY(readingStatus.getLayoutY()
+ 100);

editReadingStatus.layoutXProperty().bind(this.widthProperty().
subtract(setReadingStatus.widthProperty()).divide(2.0));

editReadingStatus.setLayoutY(setReadingStatus.getLayoutY());

removeStatus.layoutXProperty().bind(this.widthProperty().subtr
act(setReadingStatus.widthProperty()).divide(1.65));

removeStatus.setLayoutY(setReadingStatus.getLayoutY());
        readingStatus.setVisible(false);
        setReadingStatus.setVisible(false);
        editReadingStatus.setVisible(false);
        removeStatus.setVisible(false);
        this.getChildren().addAll(readingStatus,
setReadingStatus, editReadingStatus, removeStatus);
    }

    public Button getSearch() {
        return search;
    }
    public TextField getISBN() {
        return ISBN;
    }
    public Button getSetReadingStatus() {
        return setReadingStatus;
    }
    public ComboBox getReadingStatus() {
        return readingStatus;
    }
    public Label getSetReadingStatusText() {
        return setReadingStatusText;
    }
    public Button getEditReadingStatus() {
        return editReadingStatus;
    }
    public Button getRemoveStatus() {
        return removeStatus;
    }
}
```

XIV. RegistrationPane.java

```
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.text.TextAlignment;
import javafx.scene.control.CheckBox;
import javafx.scene.control.Button;
import javafx.scene.text.Font;
import javafx.scene.layout.Pane;
public class RegistrationPane extends Pane {
    private Button register = new Button("Create User!");
    private TextField name = new TextField();
    private TextField username = new TextField();
    private TextField password = new TextField();
    private CheckBox author = new CheckBox();
    private CheckBox reader = new CheckBox();
    public RegistrationPane() {
        createRegistrationPane();
    }
    public void createRegistrationPane() {
        this.setStyle("-fx-background-color: antiquewhite;");
        Label welcomeText = new Label("Welcome To");
        Font font = new Font("Century Schoolbook", 48);
        welcomeText.setFont(font);
        welcomeText.setTextAlignment(TextAlignment.CENTER);

welcomeText.layoutXProperty().bind(this.widthProperty().subtra
ct(welcomeText.widthProperty()).divide(2));

        Image image = new
Image(getClass().getResourceAsStream("BookNookLogo.png"));
        ImageView view = new ImageView();
        view.setImage(image);

view.layoutXProperty().bind(this.widthProperty().subtract(view
.getFitWidth()).divide(3));
        view.setLayoutY(welcomeText.getLayoutY() + 50);
        view.setFitHeight(500);
        view.setFitWidth(500);
        view.setSmooth(true);
        view.setCache(true);
        view.setPreserveRatio(true);
        this.getChildren().add(view);


name.layoutXProperty().bind(this.widthProperty().subtract(name
.widthProperty()).divide(2));
        name.setLayoutY(welcomeText.getLayoutY() + 200.0);

        Label nameLabel = new Label("Please enter your
preferred name:");

nameLabel.layoutXProperty().bind(this.widthProperty().subtract
(name.widthProperty()).divide(2));
        nameLabel.setLayoutY(name.getLayoutY() - 20.0);


username.layoutXProperty().bind(this.widthProperty().subtract(
username.widthProperty()).divide(2));
        username.setLayoutY(name.getLayoutY() + 75.0);

        Label usernameLabel = new Label("Please enter your
username:");

usernameLabel.layoutXProperty().bind(this.widthProperty().subt
ract(username.widthProperty()).divide(2));
```

```
        usernameLabel.setLayoutY(username.getLayoutY() -
20.0);


password.layoutXProperty().bind(this.widthProperty().subtract(
password.widthProperty()).divide(2));
        password.setLayoutY(username.getLayoutY() + 75.0);

        Label passwordLabel = new Label("Please enter your
password:");

passwordLabel.layoutXProperty().bind(this.widthProperty().subt
ract(password.widthProperty()).divide(2));
        passwordLabel.setLayoutY(password.getLayoutY() -
20.0);

        Label role = new Label("Are you an author or a
reader?");

role.layoutXProperty().bind(this.widthProperty().subtract(pass
word.widthProperty()).divide(2));
        role.setLayoutY(password.getLayoutY() + 50.0);


author.layoutXProperty().bind(this.widthProperty().subtract(pa
ssword.widthProperty()).divide(2));
        author.setLayoutY(role.getLayoutY() + 20.0);

        Label authorLabel = new Label("Author");

authorLabel.layoutXProperty().bind(this.widthProperty().subtra
ct(password.widthProperty()).divide(1.85));
        authorLabel.setLayoutY(author.getLayoutY());


reader.layoutXProperty().bind(this.widthProperty().subtract(pa
ssword.widthProperty()).divide(2));
        reader.setLayoutY(author.getLayoutY() + 20.0);

        Label readerLabel = new Label("Reader");

readerLabel.layoutXProperty().bind(this.widthProperty().subtra
ct(password.widthProperty()).divide(1.85));
        readerLabel.setLayoutY(reader.getLayoutY());



register.layoutXProperty().bind(this.widthProperty().subtract(
password.widthProperty()).divide(2));
        register.setLayoutY(reader.getLayoutY() + 75.0);

        this.getChildren().add(welcomeText);
        this.getChildren().addAll(name, nameLabel);
        this.getChildren().addAll(username, usernameLabel,
password, passwordLabel);
        this.getChildren().addAll(role, author, authorLabel,
reader, readerLabel);
```

```
        this.getChildren().add(register);
    }

    public Button getRegister() {
        return register;
    }

    public TextField getName() {
        return name;
    }

    public TextField getUsername() {
        return username;
    }

    public TextField getPassword() {
        return password;
    }

    public CheckBox getAuthor() {
        return author;
    }

    public CheckBox getReader() {
        return reader;
    }
}
```