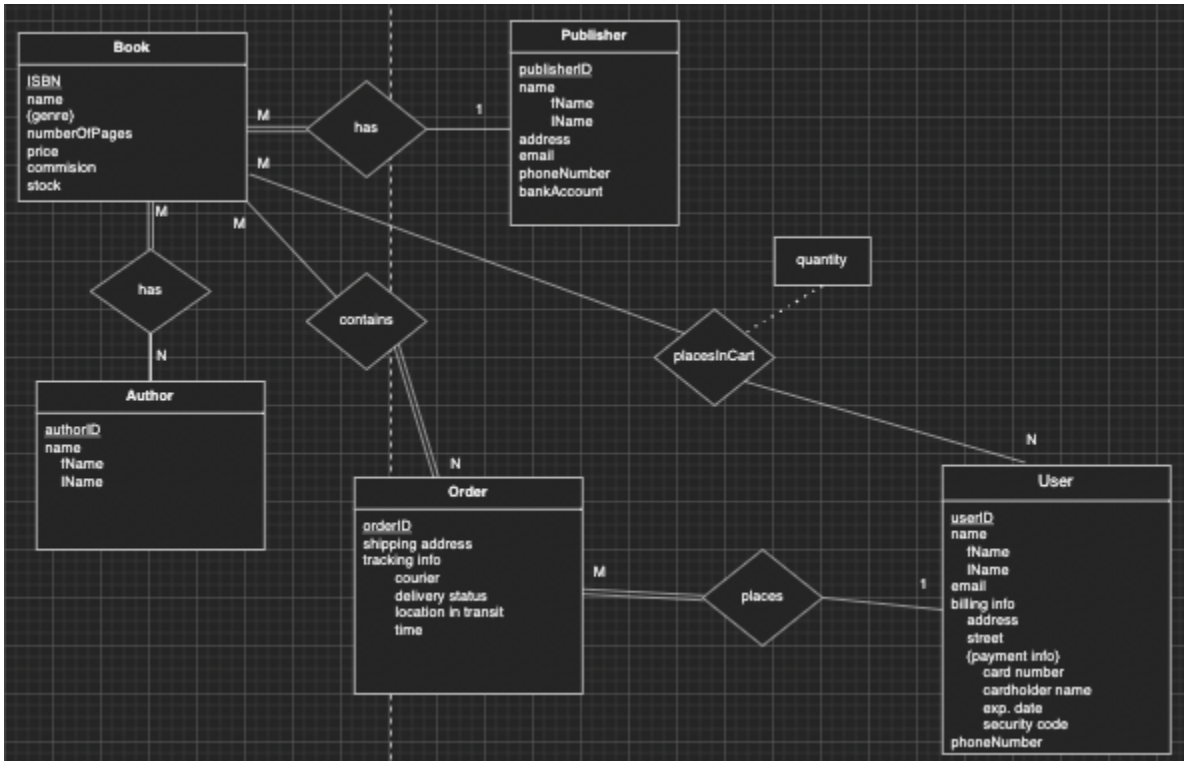
COMP 3005 Final Project Report

Peter Pham 101141273
Eric Steward 101144582

Conceptual Design

Entity Relationship Diagram



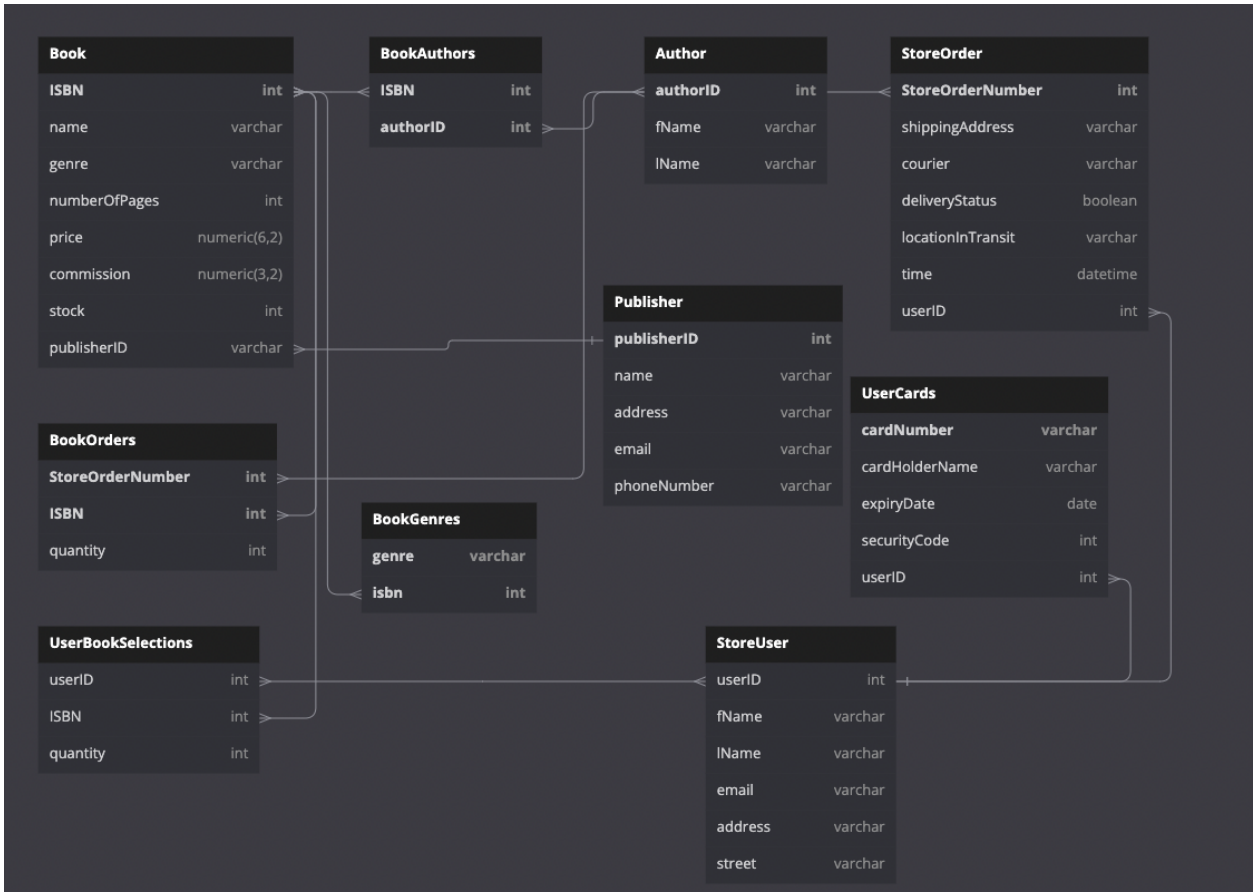
Assumptions:

- Books must have a publisher and author(s)
- Users and Publishers may only have one phone number and one email address
- Users may have multiple payment infos (cards)
- Publishers may only have one bank account

Reduction to Relational Schemas

Database Design

Primary Keys are bolded. Cardinalities are indicated by whether a connection is connected by 3 lines or 1 line. For instance, Book.publisherID -> Publisher.publisherID is a many-to-one cardinality. From the ER diagram, the placesInCart relationship is realized as a table called UserBookSelections which maps userIDs to books selected. A trigger handles removing user book selections when an order is placed.



Normalization of Relation Schemas

The reduction to database design already satisfies **Boyce Codd Normal Form** and is thus already in a **good normal form**. Observe the following Boyce Codd Normal Form Tests.

1. Define each relation:

Author(authorID,fName,lName)
Book(ISBN,name,numberOfPages,price,commission,stock,publisherID)
BookOrders(orderNumber,ISBN, BookOrders.quantity)
BookAuthors(ISBN,authorID)
BookGenres(ISBN,genre)
Order(orderNumber,shippingAddress,courier,deliveryStatus,locationInTransit,time,userID,cardNumber)
Publisher(publisherID, name, address, email, phoneNumber,bankAccountNumber)
User(UserID,fName,lName,email,address,street)
UserBookSelection(userId,ISBN, UserBookSelection.quantity)
UserCards(cardNumber, cardHolderName, expiryDate, securityCode, userID)

2. Consider all functional dependencies:

authorID → author.fName, author.lName
orderNumber,ISBN → BookOrders.quantity
userID,ISBN → UserBookSelection.quantity

userID → fName, lName, email, address, street
cardNumber →cardHolderName, expiryDate, securityCode, **userID**
publisherId → publisher.name, address, email, phoneNumber, bankAccountNumber
ISBN → name, genre, numberOfPages, price, commission, stock, **publisherID**
orderNumber → shippingAddress, courier, deliveryStatus, locationInTransit, time, **userID**

3. TestBCNF

For each functional dependency $f \in F$ where F is the set of functional dependencies that apply to the schema. Represent f as $\alpha \rightarrow \beta$. For each relation, test the functional dependencies that hold for that relation. That is, for functional dependencies where α contains no attributes of the relation being tested, ignore them.

Author(authorID,fName,lName)

- authorID → author.fName, author.lName. Since authorID determines fName and lName, it is a superkey of the relation, this relation is in BCNF.

Book(ISBN,name,numberOfPages,price,commission,stock,publisherID)

- orderNumber,ISBN → BookOrders.quantity: holds BCNF because α includes ISBN, which is a superkey of Book.
- userID,ISBN → UserBookSelection.quantity: holds BCNF because α includes ISBN, which is a superkey of Book.
- publisherId → publisher.name, address, email, phoneNumber, bankAccountNumber: holds because closure of publisherId does not contain any other attributes of Book
- ISBN → name, numberOfPages, price, commission, stock, **publisherID:** holds BCNF because α includes ISBN, which is a superkey of Book.

BookOrders(orderNumber,ISBN,quantity)

- orderNumber,ISBN → BookOrders.quantity: holds BCNF because
  $orderNumber, ISBN, BookOrders.\ quantity \subset (orderNumber, ISBN^{+})$,. Since the closure includes all attributes of BookOrders, α is a superkey and holds BCNF.
- orderNumber → shippingAddress, courier, deliveryStatus, locationInTransit, time, **userID:** holds BCNF because
  $(orderNumber)^{+} = shippingAddress,\ courier,\ deliveryStatus,\ locationInTransit,\ time,\ userID,\ fName,\ lName,\ email,\ address,\ street$

does not include any attributes of
$BookOrders - orderNumber = BookOrders.quantity, ISBN$.

- ISBN → name, genre, numberOfPages, price, commission, stock, **publisherID:** holds BCNF because
  $(ISBN)^+ = name, genre, numberOfPages, price, commission, stock, publisherID, publisher.name, address, email, phoneNumber$ which does not include any attributes of $BookOrders - ISBN = BookOrders.quantity, userID$

BookAuthors(ISBN,authorID)

- ISBN → name, genre, numberOfPages, price, commission, stock, **publisherID:** holds BCNF because
  $(ISBN)^+ = name, genre, numberOfPages, price, commission, stock, publisherID, publisher.name, address, email, phoneNumber$ which does not include any attributes of $BookAuthors - ISBN = authorID$
- authorID → author.fName, author.lName: holds BCNF because
  $(authorID)^+ = author.fName, author.lName$
  does not include any attributes of $BookAuthors - authorID = ISBN$
- userID,ISBN → UserBookSelection.quantity: holds because
  $(userID, ISBN)^+ = fName, lName, email, address, street, name, genre, numberOfPages, price, commission, stock, publisherID, name, address, email, phoneNumber, UserBookSelection.quantity$ which does not include any attributes of $BookAuthors - userID, ISBN = authorID$

BookGenres(ISBN,genre)

- ISBN → name, numberOfPages, price, commission, stock, **publisherID:** holds BCNF because
  $(ISBN)^+ = name, numberOfPages, price, commission, stock, publisherID, publisher.name, address, email, phoneNumber$ does not include any attributes of $BookGenres - ISBN = genre$
- userID,ISBN → UserBookSelection.quantity: holds because
  $(userID, ISBN)^+ = fName, lName, email, address, street, name, genre, numberOfPages, price, commission, stock, publisherID, name, address, email, phoneNumber, UserBookSelection.quantity$ does not include any attributes of $BookAuthors - userID, ISBN = authorID$

Order(orderNumber,shippingAddress,courier,deliveryStatus,locationInTransit,time, userID)

- orderNumber → shippingAddress, courier, deliveryStatus, locationInTransit, time, **userID:** holds BCNF because α includes orderNumber, which is a superkey of Order.

- userID → fName, lName, email, address, street: holds BCNF because $(userID)^+ = fName,\ lName,\ email,\ address,\ street$ does not include any attributes of $Order\ -\ userID\ =\ orderNumber, shippingAddress, courier, deliveryStatus, locationInTransit, time$
- orderNumber,ISBN → BookOrders.quantity:

Publisher(publisherID, name, address, email, phoneNumber)

- publisherId → publisher.name, address, email, phoneNumber: holds because publisherId is a superkey for Publisher

User(UserID,fName,lName,email,address,street)

- userID → fName, lName, email, address, street: holds because α includes userID, which is a superkey of User.
- userID,ISBN → UserBookSelection.quantity: holds BCNF because α includes userID, which is a superkey of User.

UserBookSelection

- userID,ISBN → UserBookSelection.quantity: holds BCNF because $userID, ISBN,\ UserBookSelection.quantity\ \subset\ (userID, ISBN)^+$. Since the closure includes all attributes of UserBookSelection, α is a superkey and holds BCNF.
- ISBN → name, genre, numberOfPages, price, commission, stock, **publisherID:** holds BCNF because $(ISBN)^+ = name,\ genre,\ numberOfPages,\ price,\ commission,\ stock,\ publisherID,\ publisher.name,\ address,\ email,\ phoneNumber$ does not include any attributes of $UserBookSelection\ -\ ISBN\ =\ UserBookSelection.quantity,\ userID$
- userID → fName, lName, email, address, street: holds BCNF because $(userID)^+ = fName,\ lName,\ email,\ address,\ street$ does not include any attributes of $UserBookSelection\ -\ userID\ =\ UserBookSelection.quantity,\ ISBN$
- orderNumber,ISBN → BookOrders.quantity: holds BCNF because $(orderNumber, ISBN)^+ = shippingAddress,\ courier,\ deliveryStatus,\ locationInTransit,\ time,\ userID,\ fName,\ lName,\ email,\ address,\ street,\ name,\ genre,\ numberOfPages,\ price,\ commission,\ stock,\ publisherID, BookOrders.quantity$ does not include any attributes of $UserBookSelection\ -\ orderNumber,\ ISBN\ =\ UserBookSelection.quantity$

UserCards(cardNumber, cardHolderName, expiryDate, securityCode, userID)

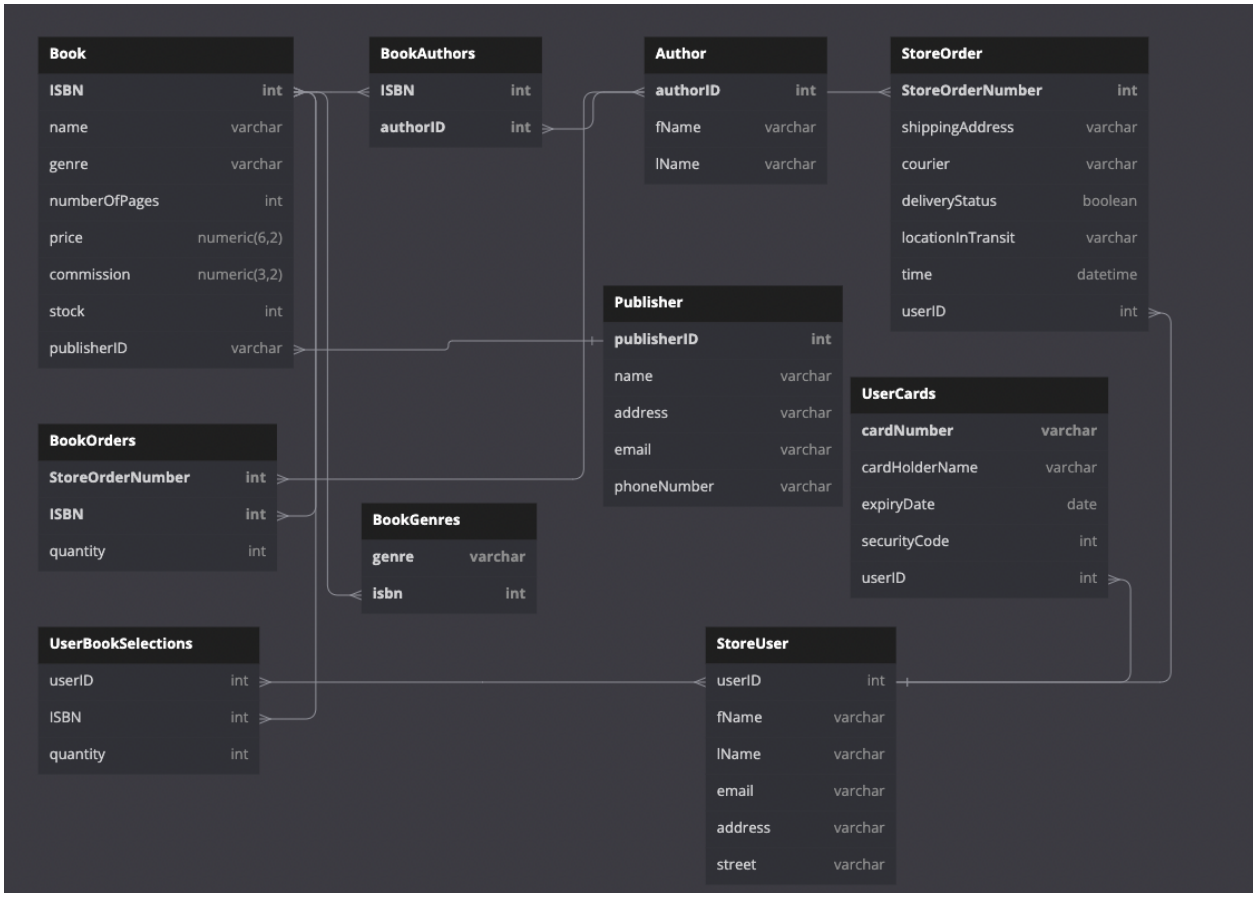
- cardNumber →cardHolderName, expiryDate, securityCode, **userID:** holds because $(cardNumber)^+ = cardHolderName,\ expiryDate,\ securityCode,\ userID, fName, lName,\ email,\ address,\ street$ which is a superkey for UserCards
- userID → fName, lName, email, address, street: holds because $(userID)^+ = fName,$

$lName, email, address, street$ does not include any attributes of $UserCards - userID = cardNumber, cardHolderName, expiryDate, securityCode$

- userID,ISBN → UserBookSelection.quantity: holds because $(userID, ISBN)^{+} = fName, lName, email, address, street, name, genre, numberOfPages, price, commission, stock, publisherID, name, address, email, phoneNumber, UserBookSelection.quantity$ does not include any attributes of $UserCards - userID, ISBN = cardNumber, cardHolderName, expiryDate, securityCode$

Database Schema Diagram

The following shows the database schema diagram with the same names used in the database. Primary keys are bolded and cardinalities are determined by connection shapes. For instance, Book.publisherID -> Publisher.publisherID is many-to-one which aligns with the constraint that a book can have at most one publisher while a publisher can publish many books.



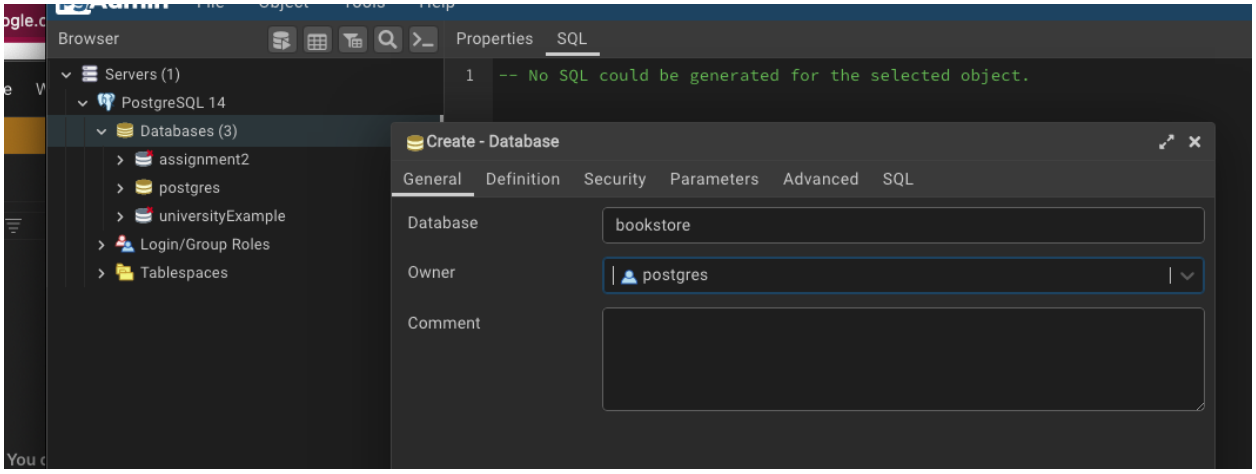
2.5 Implementation

This application uses the PERN (Postgres, Express, React, Node) to implement the bookstore. Postgres is the relational DBMS used containing all tables in third normal form. Express and Node are used for the backend to make the relevant SQL operations to manipulate the database. Finally, React is used for the web-frontend to implement a UI and communicate with the Express backend.

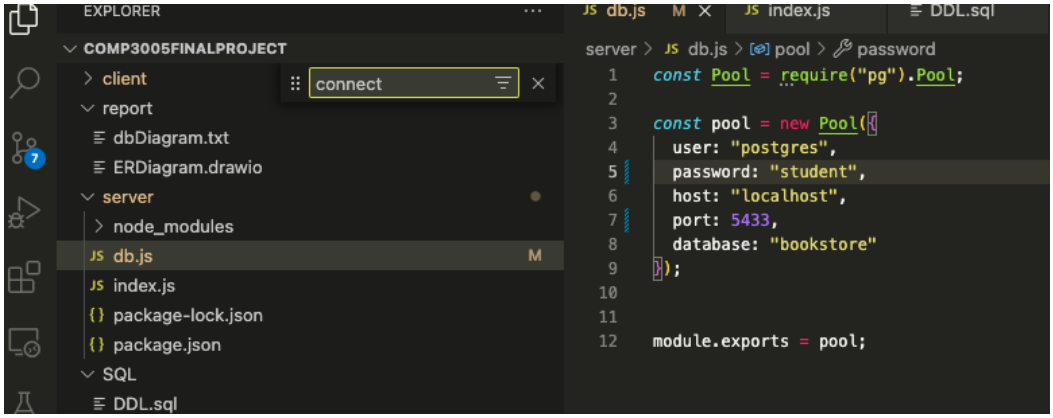
2.5.1 Getting Started

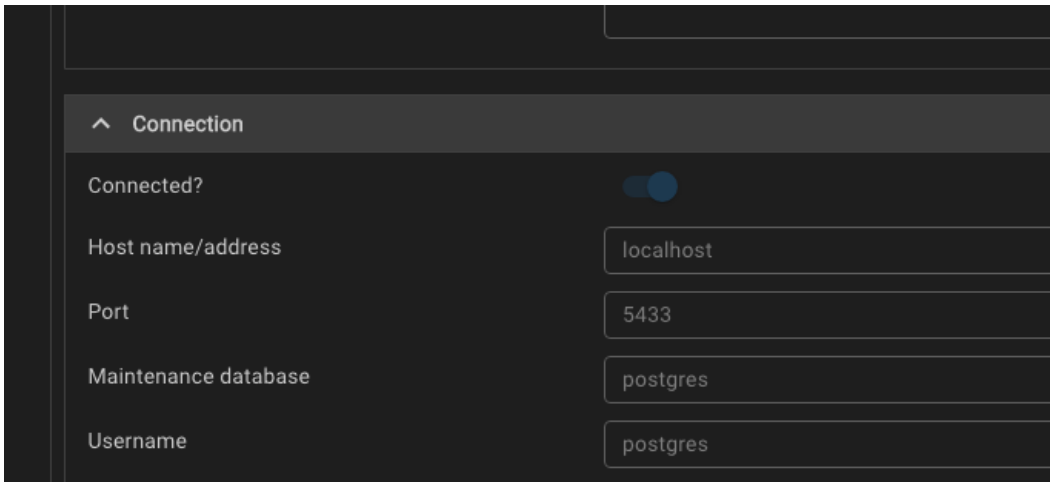
The following section indicates how to set up the program to be up and running.

1. Connect to your postgres server and create a database titled "bookstore"



2. Make sure the postgres database port and username are aligned with the port and user fields in the db.js file. Additionally, input the password used for your database connection.



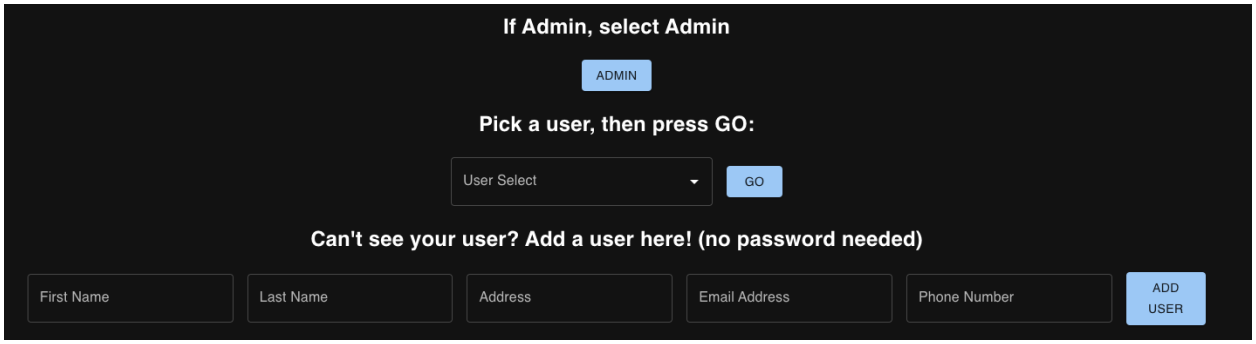


3. Using the query tool, paste in the init.sql script to add all tables, functions, triggers, and sample data.
4. Navigate to the server directory and run npm install to download dependencies.
5. Navigate to the server directory and run npm install to download dependencies.
6. In the server directory, run nodemon start
7. In the client directory, run npm run start

It is assumed that the client will be hosted on port 3000 and that the server will be hosted on port 5000. If these ports are not being used, the program will not work. This can be fixed by stopping whichever services are occupying those ports prior to running.

2.5.2 Mode Select Interface

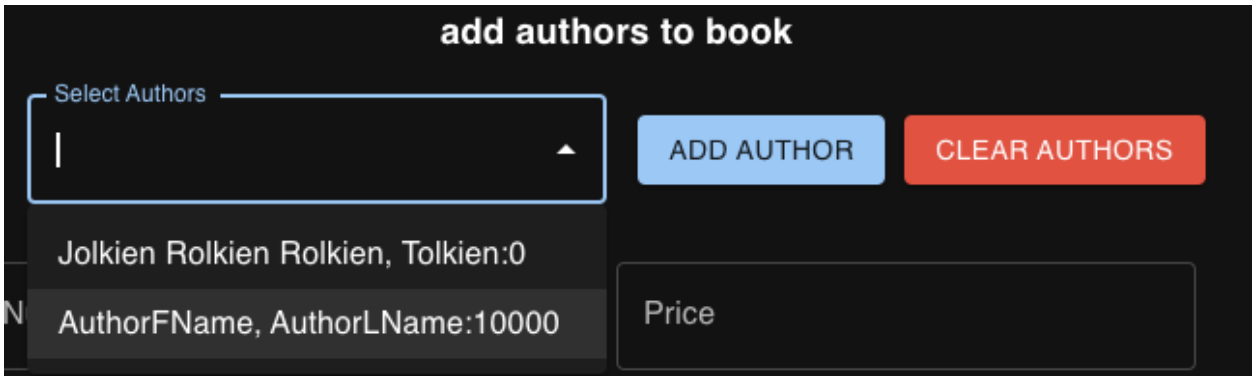
The project begins at the url "/" and offers a mode select page. In this page, the user may choose to enter the admin home or user home after selecting a user. We will first demonstrate the admin home page.



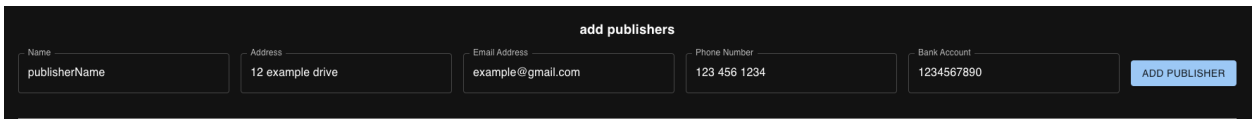
2.5.3 Admin Home

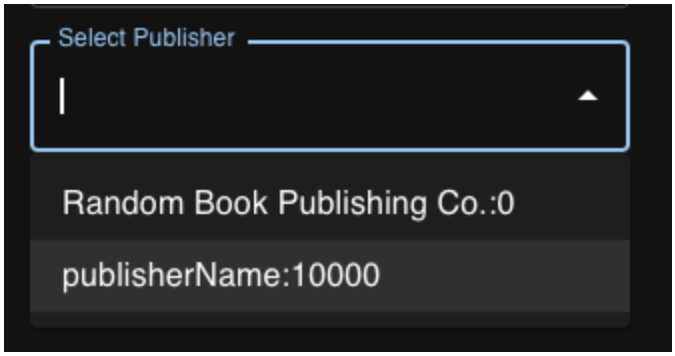
The admin page allows ordering new books, adding authors, publishers, and books. There is also a reports section showing current sales, expenditures, sales per genre, sales per author, and sales per publisher.

To add a new author, enter a first and last name and hit the add author button. Notice that the new author is seen in the select authors drop down.

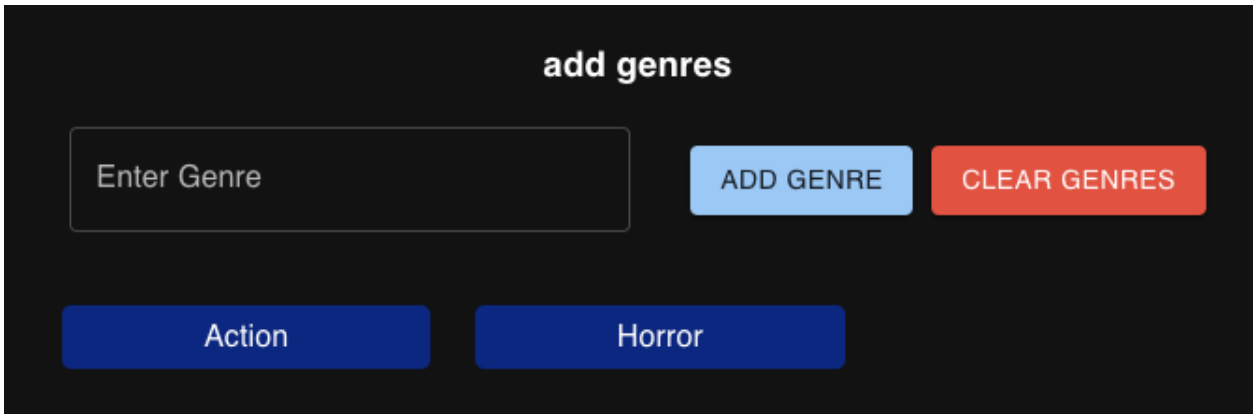


To add a new publisher, fill out the name, address, email, phone number, and bank account fields, then hit add publisher. Notice that the new publisher is seen in the select publisher drop down.

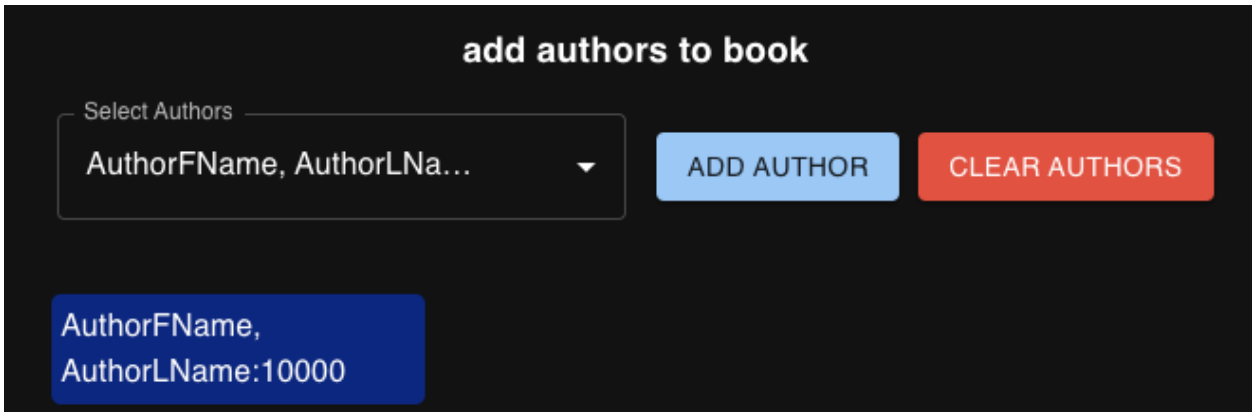




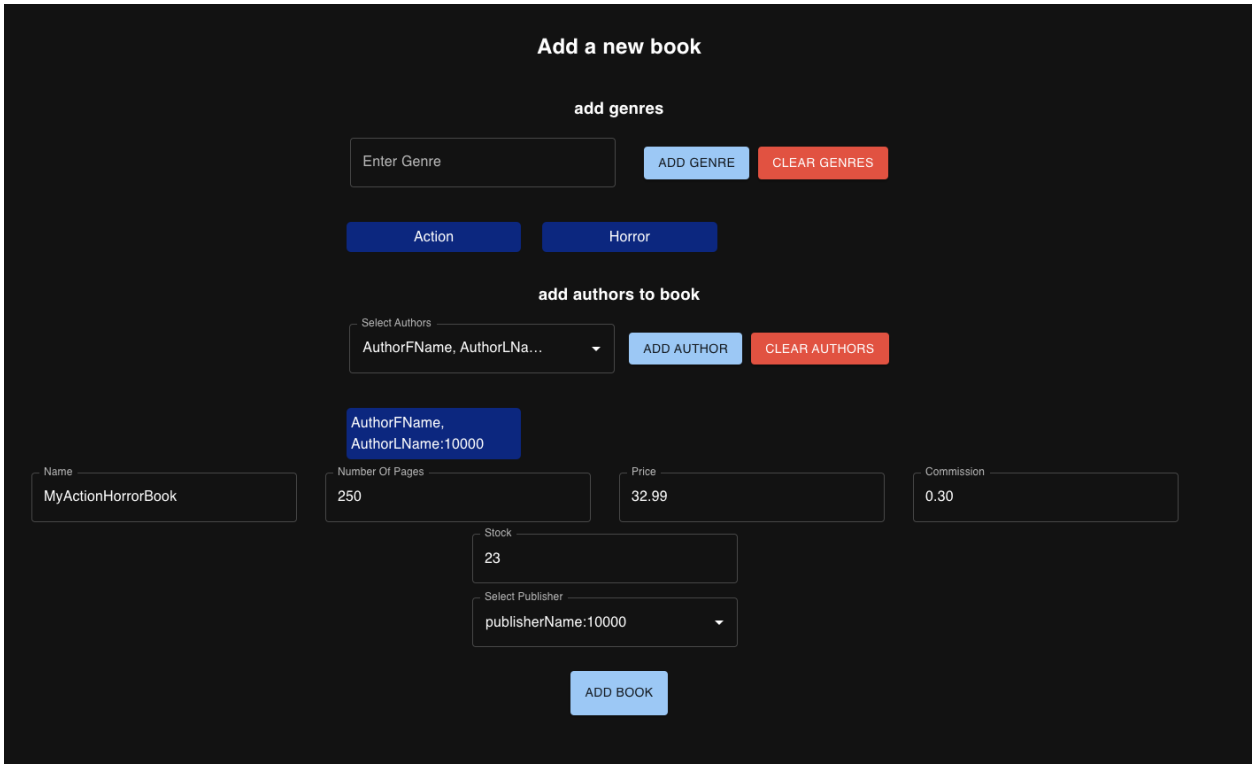
To add a book, enter in a genre and hit add genre. Repeat for as many genres as desired.



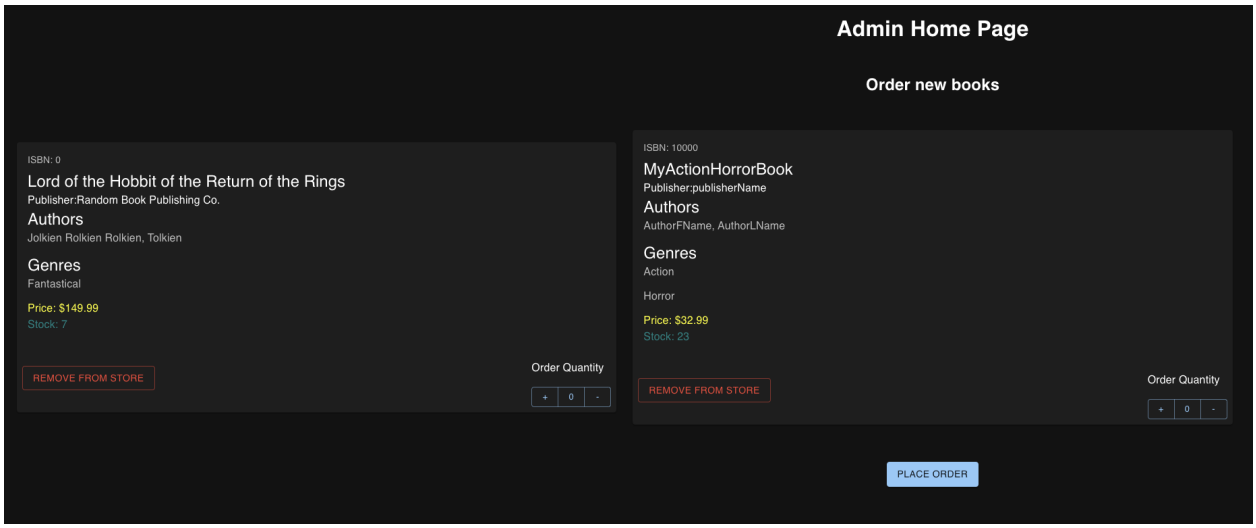
Then, select an author and hit add author. Repeat for as many authors as desired.



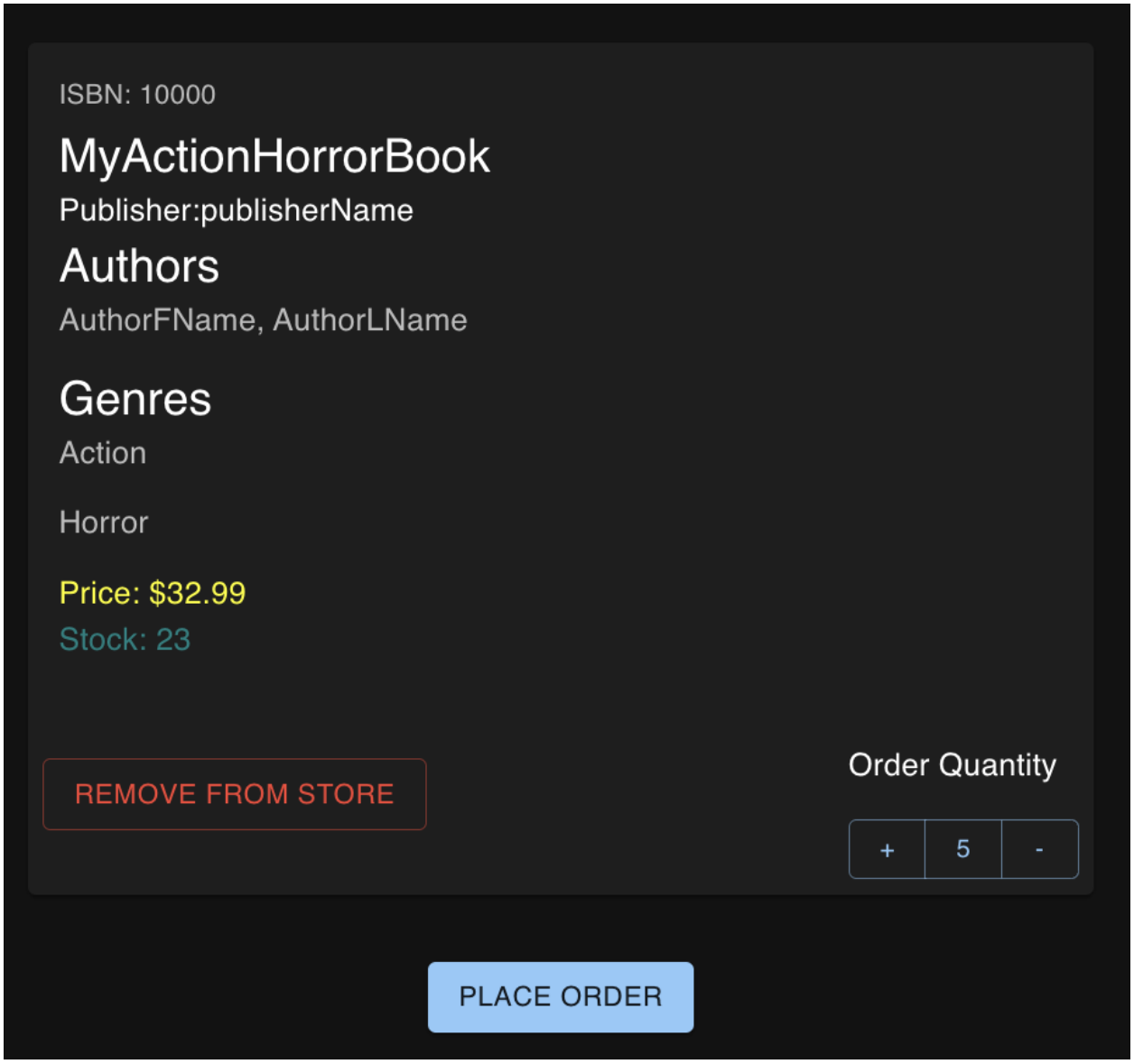
Finally, fill out the name, number of pages, price, commission, and stock values. Select a publisher from the drop down. Finally hit add book to add the new book to the store.

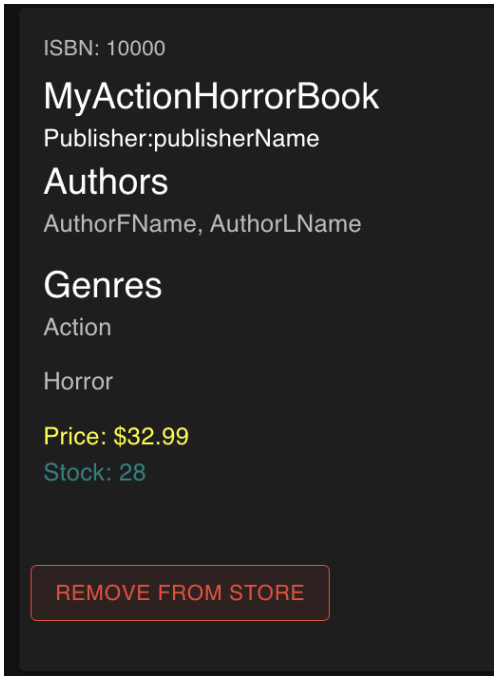


Observe that the new book has been added to the store and we can add stock to it.



Order more stock for a book by using the + and - buttons on each book card and hitting the place order button. Let's add 5 stock to the MyActionHorrorBook.


ISBN: 10000
MyActionHorrorBook
Publisher:publisherName
Authors
AuthorFName, AuthorLName
Genres
Action
Horror
Price: $32.99
Stock: 23
REMOVE FROM STORE
Order Quantity
+
5
-
PLACE ORDER



Observe that this book has increased stock. Each book card has a remove from store button that deletes it from the store. For testing purposes, do not hit this for now so we can test search functionalities. Finally, observe the report section of the admin page. These values are based on pre-loaded orders and will change once new orders are made.

Reports /

Sales Report

Total sales
449.97

Total expenditures
22.50

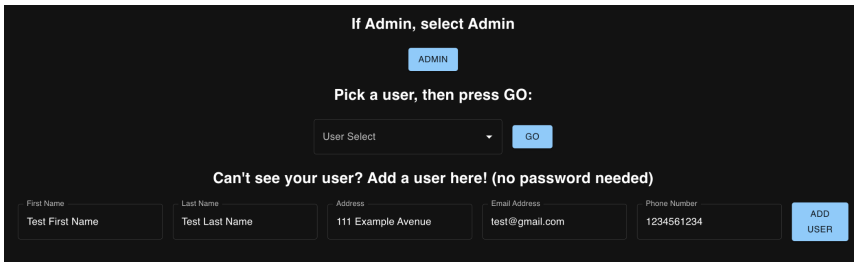
Sales per genre
Fantastical: 449.97

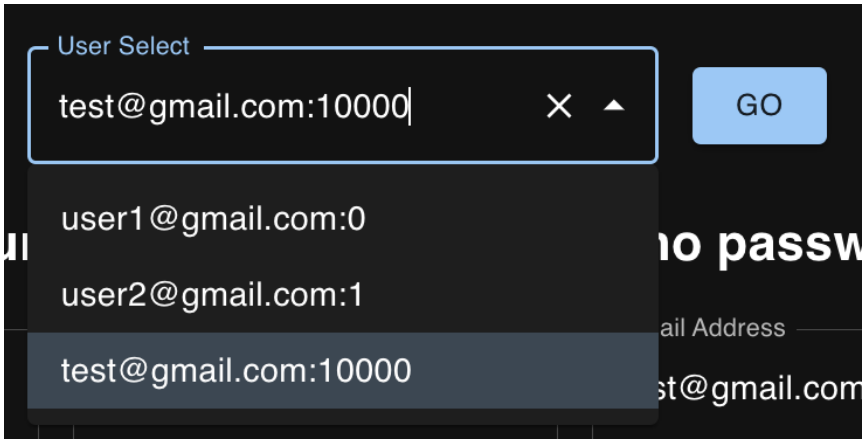
Sales per author
Jolkien Rolkien Rolkien Tolkien: 449.97

Sales per publisher
Random Book Publishing Co.: 22.50

2.5.4 User Registration

For this section, navigate back to "/" to enter the mode select page. In this page, the user may select from a list of users or add a user if they do not see their desired user. After adding a user, the new user's email will appear in the drop down with their userID at the end separated by a colon.



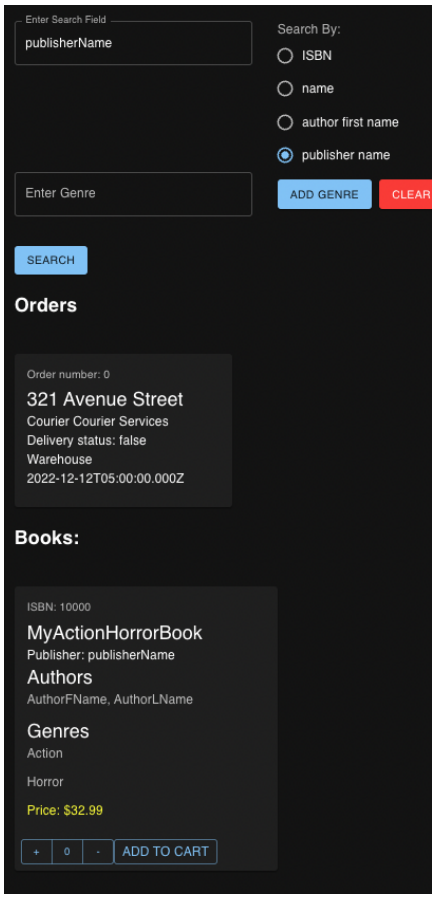


After selecting a user and pressing go, the app redirects to the /userHome url. For testing purposes, login via user1@gmail.com:0 since there is pre-loaded data.

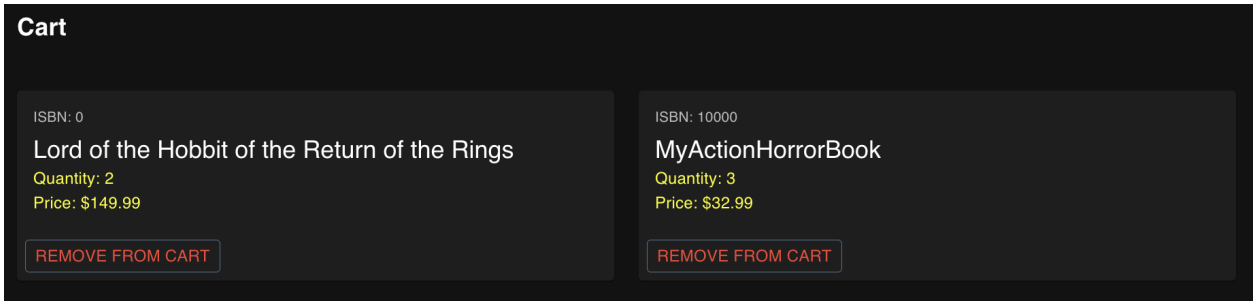
2.5.5 User Home

The user home page allows searching for books by ISBN, name, author first name, or publisher name. User Orders are also displayed. Finally there is a section to add specified quantities of books to a cart before checking out. Books can be removed from cart via a red button on the book card.

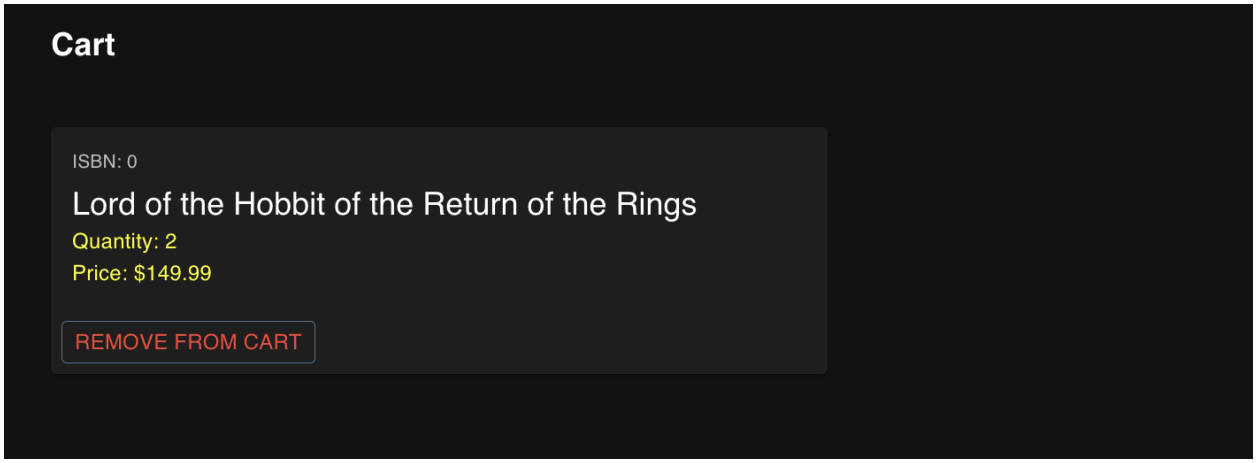
Test searching a book by entering an exact match to a field. For instance, to search by publisher name, hit the publisher name radio button. Then to test, enter in "publisherName" in the search field. Press search and observe that MyActionHorrorBook is the only book that shows. Note: search by genre does not work.



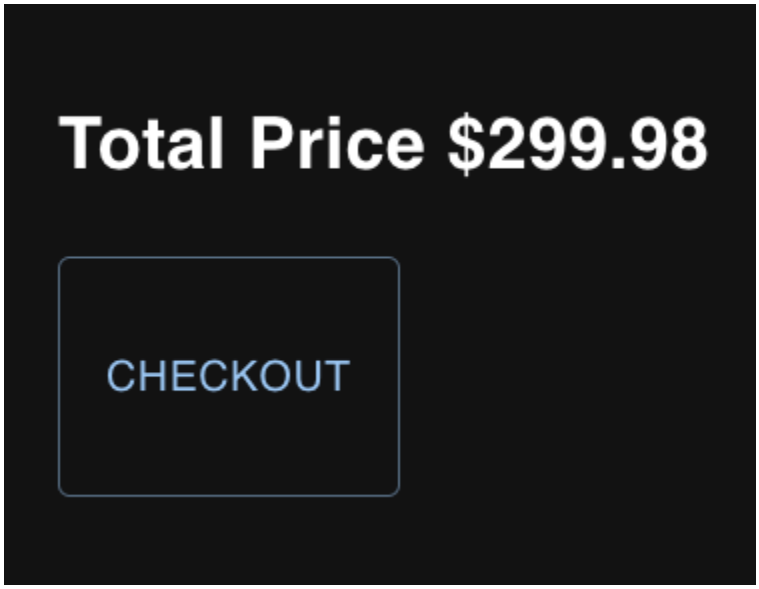
Refresh the page to view all books again. Add some book quantities using the plus and minus buttons and then clicking add to cart. Observe the books in the cart.



Try removing MyActionHorrorBook from the cart.



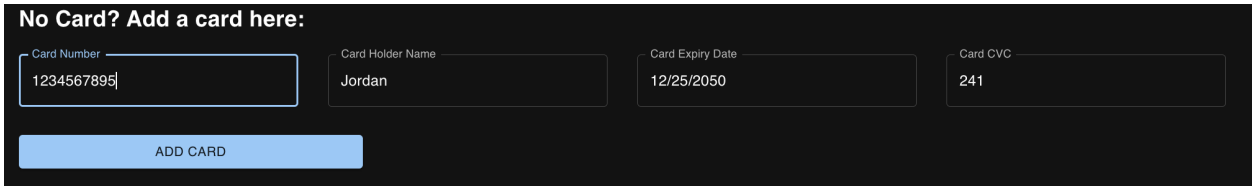
Observe the total price calculation. Then hit checkout to make an order.

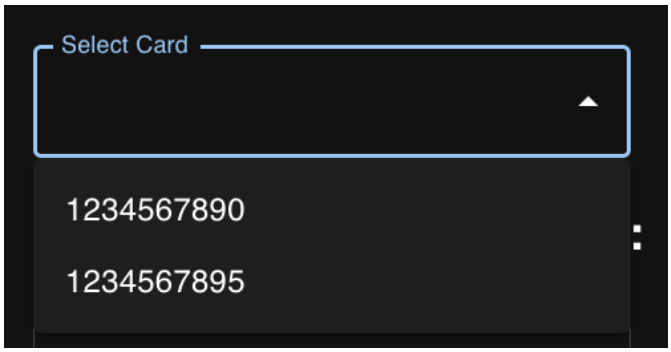


2.5.6 Checkout Page

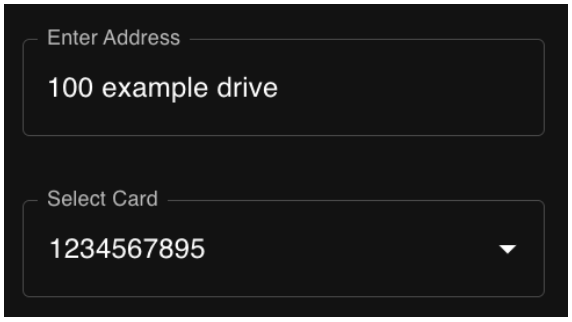
The checkout page shows the user's current book selections. The user may still remove books from the cart at this stage. If the user wants to add more books, they must navigate back using the browser back button. Before placing an order, an address, and card must be entered.

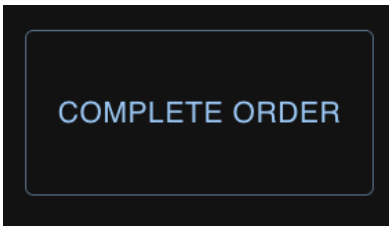
First add a card by filling out card details and hitting add card. Observe that the new card is in the select card drop down.





Select a card and enter an address. Then hit the complete order button. The user will be redirected to the userHome page where they can view the status of the new order. The admin will see the new changes in the store report.





2.8 Github Repository

See the github repository here:
https://github.com/phampe68/COMP3005FinalProject/