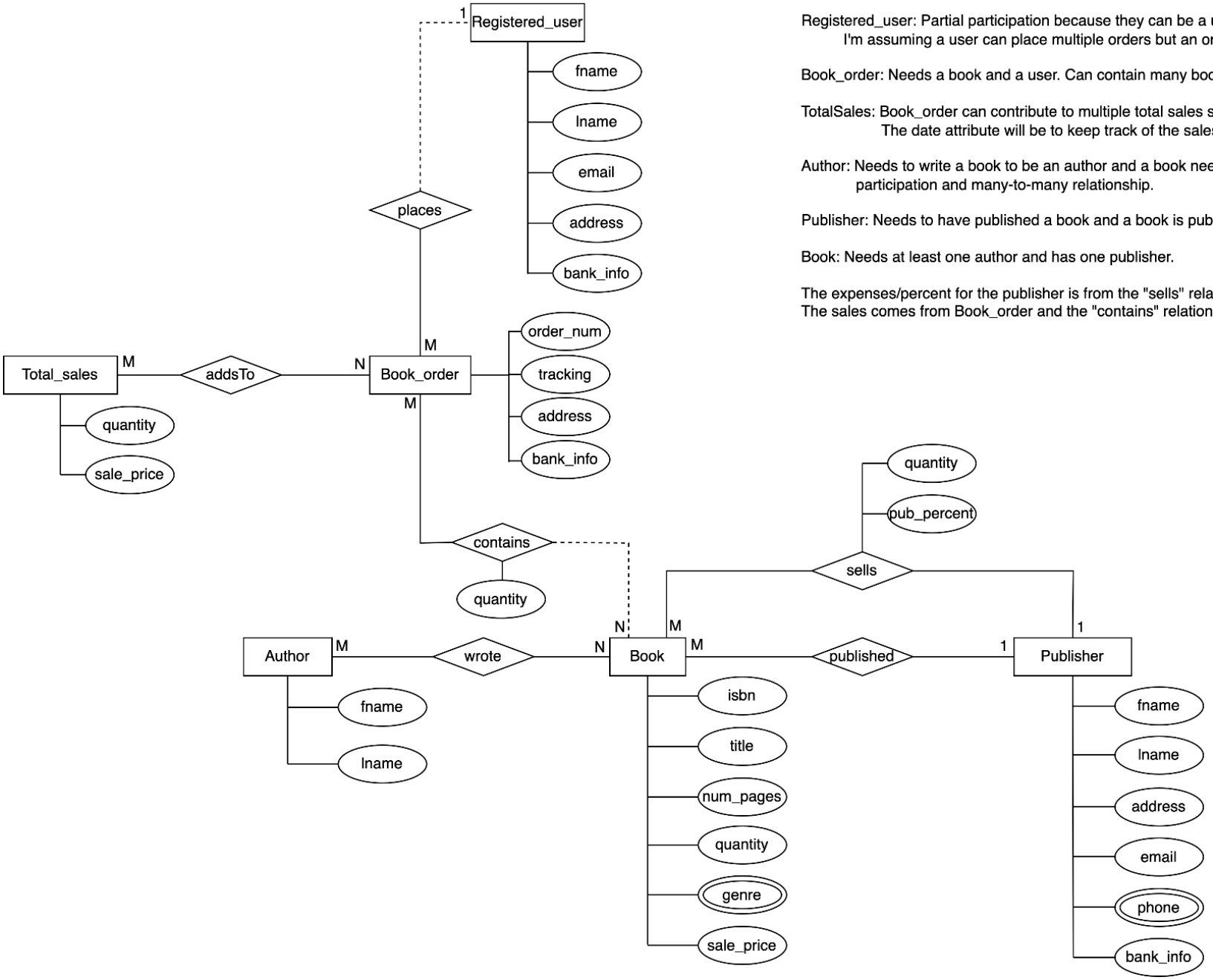
COMP 3005
Final Project
Maykala Salinas-Roy (101145833)

Look Inna Book - Project Report

Conceptual Design



Registered_user: Partial participation because they can be a user and not have an order placed.
I'm assuming a user can place multiple orders but an order belongs to only one user.

Book_order: Needs a book and a user. Can contain many books.

TotalSales: Book_order can contribute to multiple total sales since there can be multiple books in an order.
The date attribute will be to keep track of the sales by month and year.

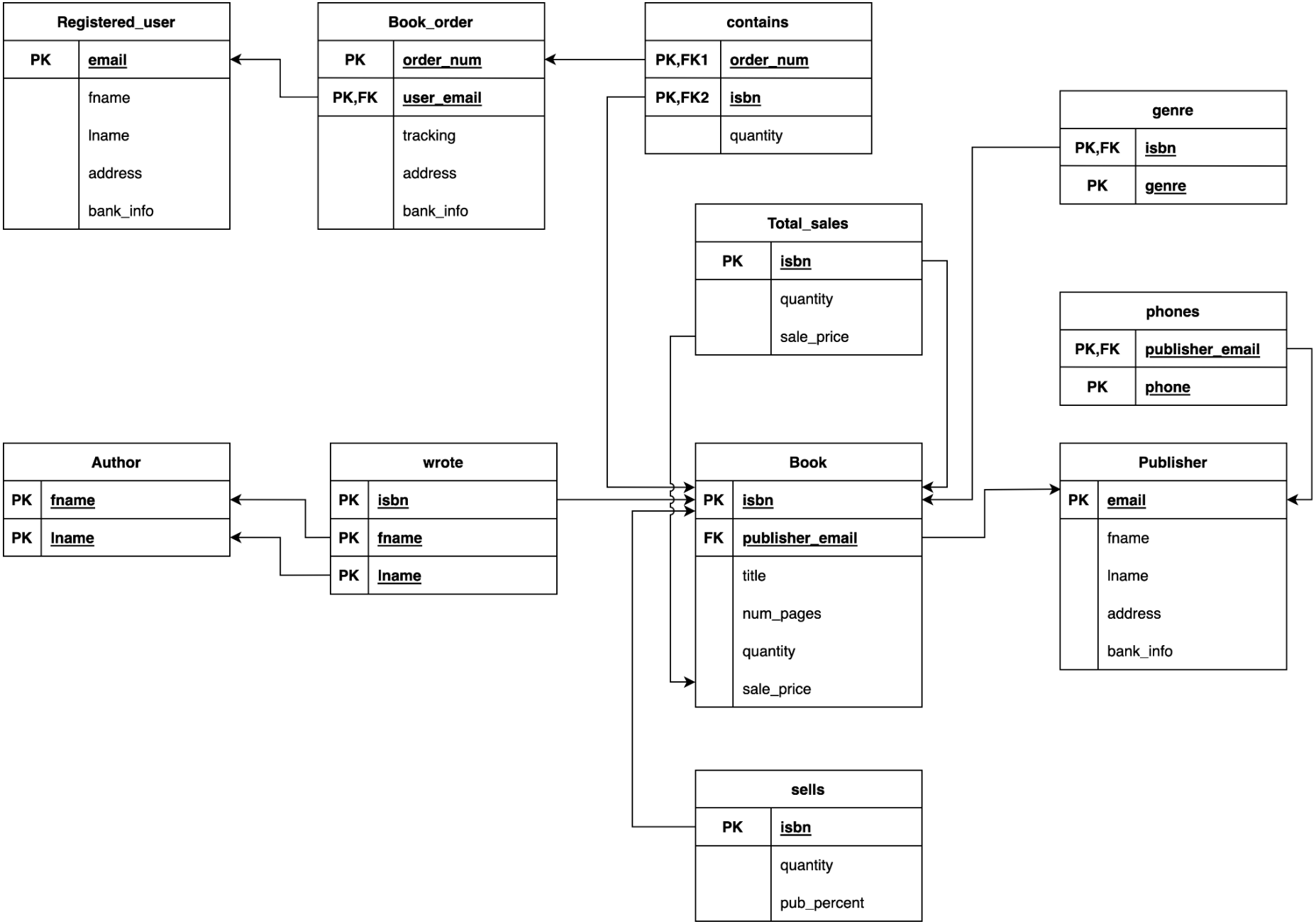
Author: Needs to write a book to be an author and a book needs at least 1 author so they are a full participation and many-to-many relationship.

Publisher: Needs to have published a book and a book is published by only one publisher.

Book: Needs at least one author and has one publisher.

The expenses/percent for the publisher is from the "sells" relationship
The sales comes from Book_order and the "contains" relationship

Relation Schemas



Normalization of Relation Schemas

I will be using the BCNF simplified test because all of my functional dependencies contain all of the attributes for their relations so they are not decompositions.

Relations with only trivial dependencies:

- Author
  - R = (fname, lname)
  - F = { fname, lname → fname, lname }
- wrote
  - R = (isbn, fname, lname)
  - F = { isbn, fname, lname → isbn, fname, lname }
- genre
  - R = (isbn, genre)
  - F = { isbn, genre → isbn, genre)
- phones
  - R = (publisher_email, phone)
  - F = { publisher_email, phone → publisher_email, phone)

All of these relations would pass since they are trivial/superkeys

Book Relation:

R = (isbn, publisher_email, title, num_pages, quantity, pub_percent, sale_price)
F = { isbn → publisher_email, title, num_pages, quantity, pub_percent, sale_price
isbn, publisher_email → title, num_pages, quantity, pub_percent, sale_price }

$(isbn)^+$
result = isbn
isbn → publisher_email, title, num_pages, quantity, pub_percent, sale_price: result = isbn, publisher_email, title, num_pages, quantity, pub_percent, sale_price
isbn, publisher_email → title, num_pages, quantity, pub_percent, sale_price: result = isbn, publisher_email, title, num_pages, quantity, pub_percent, sale_price

$(isbn)^+$ contains all of the attributes so it passes the test

$(isbn, publisher\_email)^+$
result = isbn, publisher_email
isbn → publisher_email, title, num_pages, quantity, pub_percent, sale_price: result = isbn, publisher_email, title, num_pages, quantity, pub_percent, sale_price
isbn, publisher_email → title, num_pages, quantity, pub_percent, sale_price: result = isbn, publisher_email, title, num_pages, quantity, pub_percent, sale_price

$(isbn, publisher\_email)^+$ contains all of the attributes so it passes the test as well
Therefore, the Book relation is in normal form.

Publisher Relation:
R = (email, fname, lname, address, bank_info)
F = { email → fname, lname, address, bank_info
bank_info → email, fname, lname, address }

$(email)^+$
result = email
email → fname, lname, address, bank_info: result = email, fname, lname, address, bank_info
bank_info → email, fname, lname, address: result = email, fname, lname, address, bank_info

$(email)^+$ contains all of the attributes so it passes the test
$(bank\_info)^+$
result = bank_info
email → fname, lname, address, bank_info: result = bank_info
bank_info → email, fname, lname, address: result = email, fname, lname, address, bank_info

$(bank\_info)^+$ contains all of the attributes so it passes the test
Therefore, the Publisher relation is in normal form.

Registered_user Relation:
R = (email, fname, lname, address, bank_info)
F = { email → fname, lname, address, bank_info
bank_info → email, fname, lname, address }

$(email)^+$
result = email
email → fname, lname, address, bank_info: result = fname, lname, address, bank_info
bank_info → email, fname, lname, address: result = fname, lname, address, bank_info, email

$(email)^+$ contains all of the attributes so it passes the test

$(bank\_info)^+$
result = bank_info
email → fname, lname, address, bank_info: result = bank_info
bank_info → email, fname, lname, address: result = bank_info, email, fname, lname, address

$(bank\_info)^+$ contains all of the attributes so it passes the test
Therefore, the Registered_user relation is in normal form.

Book_order Relation:
R = (order_num, user_email, tracking)
F = { order_num → user_email, tracking
order_num, user_email → tracking }

(order_num)$^+$
result = order_num
order_num → user_email, tracking : result = order_num, user_email, tracking
order_num, user_email → tracking : result = order_num, user_email, tracking

(order_num)$^+$ contains all of the attributes so it passes the test

(order_num, user_email)$^+$
result = order_num, user_email
order_num → user_email, tracking : result = order_num, user_email, tracking
order_num, user_email → tracking : result = order_num, user_email, tracking

(order_num, user_email)$^+$ contains all of the attributes so it passes the test
Therefore, the Book_order relation is in normal form.

Contains Relation:
R = (order_num, isbn, quantity)
F = { order_num, isbn → quantity }

(order_num, isbn)$^+$
result = order_num, isbn
order_num, isbn → quantity: result = order_num, isbn, quantity

(order_num, isbn)$^+$ contains all of the attributes so it passes the test
Therefore, the contains relation is in normal form.

Total_sales Relation:
R = (isbn, quantity, sale_price)
F = { isbn, → quantity, sale_price }

(isbn,)$^+$
result = isbn
isbn → quantity, sale_price: result = isbn, quantity, sale_price

(isbn)$^+$ contains all of the attributes so it passes the test
Therefore, the total_sales relation is in normal form.

The sells relation is the same as the total_sales for this test (Except it has pub_percent instead of sale_price)

Implementation

For my project, I am using Python 3.9.10 with sqlite3 for the database and PyQt5 for the GUI. Sqlite3 is part of the Python standard library so you just need to have the Python version and PyQt5.

When you are ready to run the application, you can create a fresh copy of the database if you want. To do that, run the Python file “finalProf_makingDB.py” to drop the old database and create a new one. Then run the Python file “loadData.py” which will load the database with mock data.

The automatic purchase of books from the publisher is just when the book’s quantity is less than 10. It gets checked after there is a purchase of that book.

Mock Data

I got most of the mock data from https://www.mockaroo.com/ but for some of them I just did it myself.

The mock data doesn’t have any sales so if you run any reports it won’t show much. If you want to view any of the data, they are in individual text files and in a spreadsheet for easier viewing.

For the isbns, I have them from 12-345-671-11 to 12-345-671-21 and 12-345-678-11 to 12-345-678-99. To make it easier to test I would normally use simple isbns like 11-111-111-11. When inputting in the text boxes, you will need to follow the format XX-XXX-XXX-XX.
For other string inputs, I didn’t do any input checking so they are case sensitive.

Running the Application

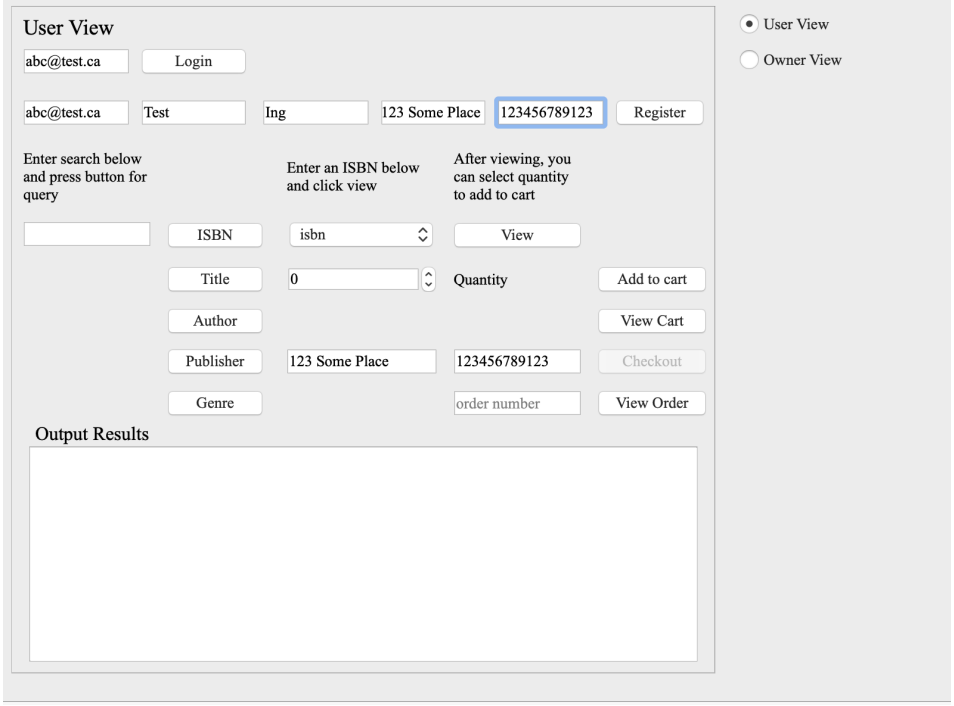
To run the application, run the Python file “bookstoreApplication.py” (‘python3 bookstoreApplication.py’ or ‘python bookstoreApplication.py’ if you only have the specified version installed) and the GUI should show.

My program starts running as a user. To switch to the owner view, click the “Owner View” radio button. To go back to the user view, click the “User View” radio button.

User View

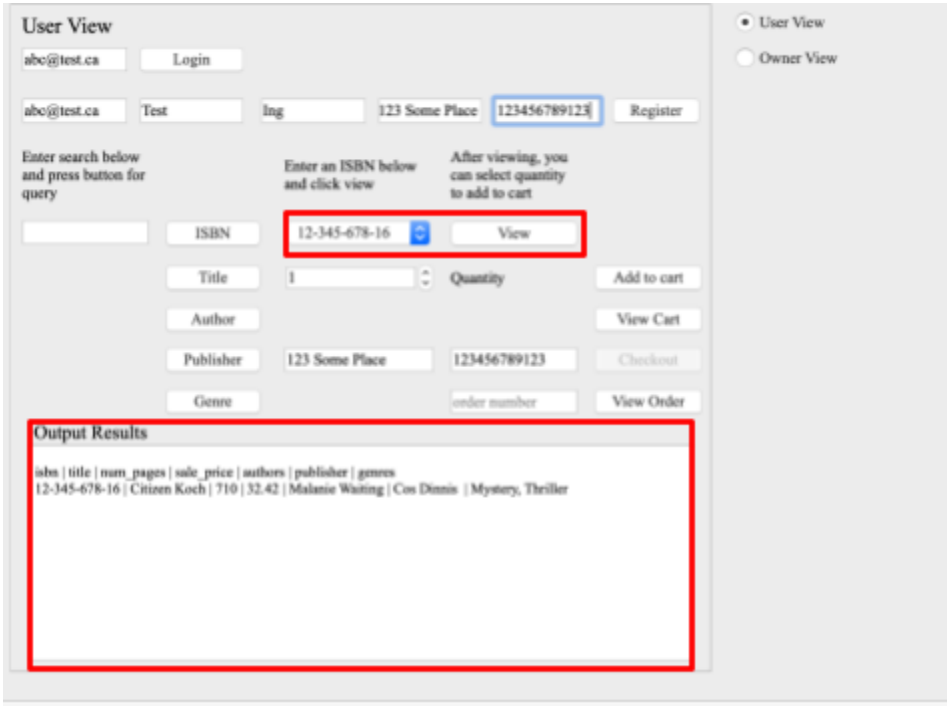
- Logging in
    - Enter your registered email in the box and press “Login”

- Registering
    - Once registered I have it set to log you in as well



- View a book

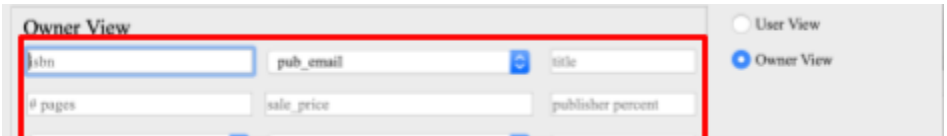
- Select an isbn from the dropdown
- Output will be in the “Output Results” (all relevant information for the book)



- Search for books
  - Type search in the box and press the corresponding button
  - Output will be in the “Output Results” (will show isbn and title)
- Add a book to cart
  - You have to view the book first so that the quantity dropdown will be populated
  - Select quantity from the dropdown and press “Add to cart”
- View cart
  - Press “View Cart”
  - Output will be in the “Output Results” (will show isbn and title)
- Checkout
  - When there is a book in the cart and you are signed in, you can checkout
  - Press “Checkout”
  - The order number will be displayed in “Output Results”
- View Order
  - Enter the valid order number in the box
  - Press “View Order”
  - Output will be in the “Output Results” (will show order number and tracking followed by the isbn and quantity of the books in the order)

Owner View

- Add a book
  - Input the necessary data and select from the dropdowns
  - Press “Add Book”



- Delete a book
    - Enter the isbn and press “Delete Book”
- Order a more of a book
    - Enter the isbn and how many to order. Then press “Order Book”
- View Reports
    - Output for these reports will be in the “Output Results”
    - “Sales vs Expenses”
        - Shows the total profit vs the total sales
        - Shows how much(quantity) of each book is sold, the dollar amount of this quantity, how much(quantity) of each book was bought from the publisher, how much was paid to the publisher
    - “Sales Per Author”
        - Shows how many(quantity) books from each author sold
    - “Sales Per Genre”
        - Shows how many(quantity) books from each genre sold
    - “Sales Per Publisher”
        - Shows how many(quantity) books from each publisher sold

Github Repository

https://github.com/maykalasalinas-roy/comp3005_finalProject