COMP 3005: Database Management Systems **(Due: Dec. 10th, 2021 (11:59 PM))**

COMP3005 Final Project Report

Instructor: Ahmed El-Roby *Name: , ID:*

Introduction

This is the project report for the COMP3005A Final Project for the Fall 2021 term.
The group for this project consists of the following members...

Group Members

- Aaron Buitenwerf (101106637)
- Hadi Cheaito (101110188)
- Nabeel Warsalee (101103167)

All project files and source code can be found at the following Github repository...

1 Conceptual Design

For the design of this database, we took a straightforward approach and attempted to keep it as simple as we could. We created four main entities that compose the bulk of what the program requires. There are then relationships and minor entities that help provide more information for our main entities while also avoiding redundancy.

The book entity contains all the information regarding the books in our system. The primary key for this entity is the isbn number. The book entity forms relationships with the publisher entity, via the relationship of a book being published by a given publisher and is also connected to the order relation via the relationship books_in_order.

The publisher entity contains all the information regarding the publishers in our system. The primary key for this entity is the name of the publisher. The publisher entity forms relationships with the book entity as mentioned earlier and is also connected to the bank_account relation via the relationship account_of.

The order entity contains all the information regarding the book orders in our system. The primary key for this entity is the order id. The order entity forms relationships with the book entity as mentioned earlier via the books_in_order relationship and is also connected to the customer relation via the relationship order_of.// Finally there is the customer entity which contains all the information regarding the customers in our system. The primary key for this entity is the email of the user. The customer entity forms relationships with the order entity as mentioned earlier via the order_of relationship.

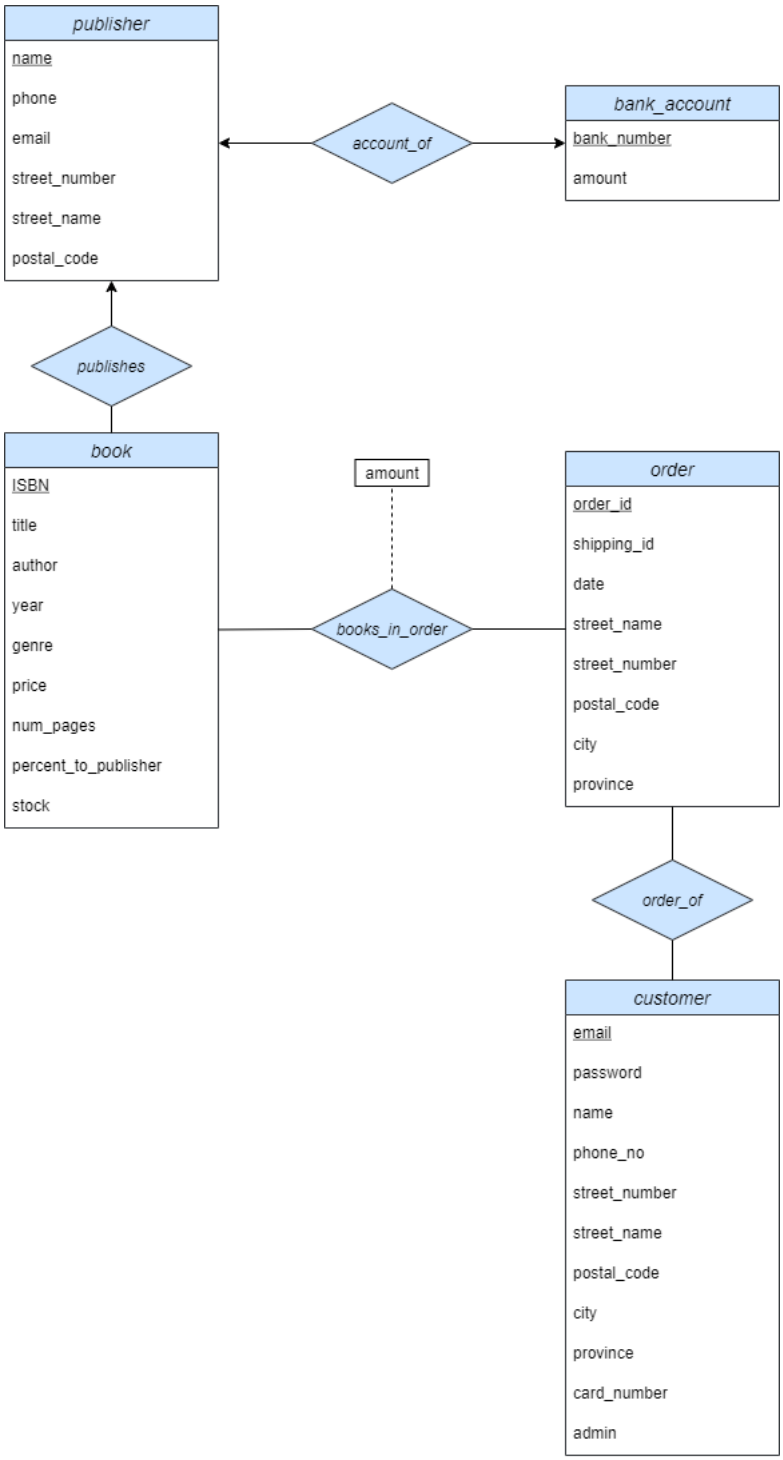
Assumptions Made
In this section we list all the assumptions that were made for certain aspects of the problem statement. These assumptions reflect how we designed and organized our database for this project.

- A book can only have one publisher
- All books with same title have the same ISBN
- Assume an order can only have one of the same book (i.e., user cannot buy two copies of the same book in one order)
- Each publisher has only one bank account

- There is only one set of reports made per publisher
- Tracking information for an order is just its shipping id

ER Diagram

The following is the Entity-Relationship Diagram created to model the entities and relationships from the provided problem statement using the assumptions we have outlined above.



2 Reduction to Relation Schemas

Here are the relation schemas gained from reducing our ER diagram into relations... (Note: Primary keys are underlined)

book(isbn, title, author, genre, year, price, num_pages, publisher_name, stock, percent_to_publisher)
order(order_id, email, shipping_id, date, street_number, street_name, postal_code, city, province)
books_in_order(order_id, isbn, amount)
customer(email, password, name, phone, street_number, street_name, postal_code, city, province, card_number, admin)
publisher(name, phone, bank_number, email, street_number, street_name, postal_code)
bank_account(bank_number, amount, debt_amount)

3 Normalization of Relation Schemas

Functional Dependencies

book
isbn → title, author, genre, year, price, num_pages, publisher_name, stock, percent_to_publisher
title, author → isbn, genre, year, price, num_pages, publisher_name, stock, percent_to_publisher

order
order_id → email, shipping_id, date, street_number, street_name, postal_code, city, province
shipping_id → email, order_id, date, street_number, street_name, postal_code, city, province
postal_code → city, province

books_in_order
order_id, isbn → amount

customer
email, password → name, phone, street_number, street_name, postal_code, city, province, card_number, admin
postal_code → city, province

publisher
name → email, phone, bank_number, street_number, street_name, postal_code
email → name, phone, bank_number, street_number, street_name, postal_code

bank_account
bank_number → amount, debt_amount

Good Normal Form Check and Decomposition

book

1st FD...
Closure of *isbn*, (*isbn*)+ = (isbn, title, author, genre, year, price, num_pages, publisher_name, stock, percent_to_publisher)
Since the closure of isbn includes all attributes in the relation, it means isbn is a superkey for the relation and it complies with BCNF.

2nd FD...
Closure of *title, author*, (*title, author*)+ = (isbn, title, author, genre, year, price, num_pages, publisher_name, stock, percent_to_publisher)
Since the closure of (title, author) includes all attributes in the relation, it means (title, author) is a superkey for the relation and it complies with BCNF.

Since all FD's for this relation satisfy the conditions for BCNF, this table is already in BCNF. Since the table is already in BCNF and no decomposition was done, all dependencies are preserved.

order

1st FD...

Closure of *order_id*, (*order_id*)+ = (order_id, email, shipping_id, date, street_number, street_name, postal_code, city, province)

Since the closure of order_id includes all attributes in the relation, it means order_id is a superkey for the relation and it complies with BCNF.

2nd FD...

Closure of *shipping_id*, (*shipping_id*)+ = (order_id, email, shipping_id, date, street_number, street_name, postal_code, city, province)

Since the closure of shipping_id includes all attributes in the relation, it means shipping_id is a superkey for the relation and it complies with BCNF.

3rd FD...

Closure of *postal_code*, (*postal_code*)+ = (postal_code, city, province)

Since the closure of postal_code does not include all attributes in the original relation, it means postal_code is not a superkey for the relation and thus violates the conditions for BCNF. We will need to decompose this relation.

Decomposition

Decompose into two new relations, order and region_order...

order(order_id, email, shipping id, date, street_number, street_name)

region_order(postal_code, city, province)

Now none of the functional dependencies violates BCNF since postal_code is now a superkey for the relation called *region_order*.

Dependency preservation

1. Check FD: order_id → email, shipping_id, date, street_number, street_name, postal_code, city, province
   Start with result r = order_id

   R_1 =(order_id, email, shipping_id, date, street_number, street_name, postal_code)
   $t = (result \cap R_1) + \cap R_1$
   t = (order_id, email, shipping_id, date, street_number, street_name, postal_code)
   result = (order_id) ∪ (order_id, email, shipping_id, date, street_number, street_name, postal_code)
   result = (order_id, email, shipping_id, date, street_number, street_name, postal_code)

   R_2 = (postal_code, city, province)
   $t = (result \cap R_2) + \cap R_2$
   t = (postal_code, province, city)
   result = (order_id, email, shipping_id, date, street_number, street_name, postal_code) ∪ (postal_code, province, city)
   result = (order_id, email, shipping_id, date, street_number, street_name, postal_code)

   Since result contains everything on the RHS of this FD, this dependency is preserved.

2. Check FD: shipping_id → email, order_id, date, street_number, street_name, postal_code, city, province
   Start with result r = shipping_id

   R_1 =(order_id, email, shipping_id, date, street_number, street_name, postal_code)
   $t = (result \cap R_1) + \cap R_1$
   t = (order_id, email, shipping_id, date, street_number, street_name, postal_code)

result = (shipping_id) ∪ (order_id, email, shipping_id, date, street_number, street_name, postal_code)
result = (order_id, email, shipping_id, date, street_number, street_name, postal_code)

R_2 = (postal_code, city, province)
$t = (result \cap R_2) + \cap R_2$
t = (postal_code, province, city)
result = (order_id, email, shipping_id, date, street_number, street_name, postal_code) ∪ (postal_code, province, city)
result = (order_id, email, shipping_id, date, street_number, street_name, postal_code)

Since result contains everything on the RHS of this FD, this dependency is preserved.

3. Check FD: postal_code → city, province
Start with result r = postal_code

R_1 =(order_id, email, shipping_id, date, street_number, street_name, postal_code)
$t = (result \cap R_1) + \cap R_1$
t = (postal_code)
result = (postal_code) ∪ (postal_code)
result = (postal_code)

R_2 = (postal_code, city, province)
$t = (result \cap R_2) + \cap R_2$
t = (postal_code, province, city)
result = (postal_code) ∪ (postal_code, province, city)
result = (postal_code, province, city)

Since result contains everything on the RHS of this FD, this dependency is preserved.

All three dependencies were shown to be preserved after running the dependency preservation algorithm, therefore the decomposition into BCNF for new relations book and region_order is dependency preserving.

books_in_order

1st FD...
Closure of *isbn, order_id*, (*isbn, order_id*)+ = (isbn, order_id, amount)
Since the closure of isbn, order_id includes all attributes in the relation, it means isbn, order_id is a superkey for the relation and it complies with BCNF.

books_in_order

1st FD...
Closure of *isbn, order_id*, (*isbn, order_id*)+ = (isbn, order_id, amount)
Since the closure of isbn, order_id includes all attributes in the relation, it means isbn, order_id is a superkey for the relation and it complies with BCNF.

customer

1st FD...
Closure of *email, password*, (*email, password*)+ = (email, password, name, phone, street_number, street_name, postal_code, city, province, card_number, admin)
Since the closure of email, password includes all attributes in the relation, it means email, password is a superkey for the relation and it complies with BCNF.

2nd FD...
Closure of *postal_code*, (*postal_code*)+ = (postal_code, city, province)
Since the closure of postal_code does not include all attributes in the original relation, it means postal_code

is not a superkey for the relation and thus violates the conditions for BCNF. We will need to decompose this relation.

Decomposition
Decompose into two new relations, order and region_order...
customer(email, password, name, phone, street_number, street_name, postal_code, card_number, admin)
region_customer(postal_code, city, province)

Now none of the functional dependencies violates BCNF since postal_code is now a superkey for the relation called *region_customer*.

Dependency preservation

1. Check FD: email, password → name, phone, street_number, street_name, postal_code, city, province, card_number, admin
   Start with result r = (email, password)

   R_1 =(email, password, name, phone, street_number, street_name, postal_code, card_number, admin)
   $t = (result \cap R_1) + \cap R_1$
   t = (email, password, name, phone, street_number, street_name, postal_code, card_number, admin)
   result = (email, password) ∪ (email, password, name, phone, street_number, street_name, postal_code, card_number, admin)
   result = (email, password, name, phone, street_number, street_name, postal_code, card_number, admin)

   R_2 = (postal_code, city, province)
   $t = (result \cap R_2) + \cap R_2$
   t = (postal_code, province, city)
   result = (email, password, name, phone, street_number, street_name, postal_code, card_number, admin)
   ∪ (postal_code, province, city)
   result = (email, password, name, phone, street_number, street_name, postal_code, card_number, admin)

   Since result contains everything on the RHS of this FD, this dependency is preserved.

2. Check FD: postal_code → city, province
   Start with result r = postal_code

   R_1 =(email, password, name, phone, street_number, street_name, postal_code, card_number, admin)
   $t = (result \cap R_1) + \cap R_1$
   t = (postal_code)
   result = (postal_code) ∪ (postal_code)
   result = (postal_code)

   R_2 = (postal_code, city, province)
   $t = (result \cap R_2) + \cap R_2$
   t = (postal_code, province, city)
   result = (postal_code) ∪ (postal_code, province, city)
   result = (postal_code, province, city)

   Since result contains everything on the RHS of this FD, this dependency is preserved.

Both dependencies were shown to be preserved after running the dependency preservation algorithm, therefore the decomposition into BCNF for new relations book and region_order is dependency preserving.

publisher

1st FD...
Closure of *name*, (*name*)+ = (name, phone, bank_number, email, street_number, street_name, postal_code)

Since the closure of name includes all attributes in the relation, it means name is a superkey for the relation and it complies with BCNF.

2nd FD...

Closure of *email*, (*email*)+ = (name, phone, bank_number, email, street_number, street_name, postal_code)

Since the closure of email includes all attributes in the relation, it means email is a superkey for the relation and it complies with BCNF.

Since all FD's for this relation satisfy the conditions for BCNF, this table is already in BCNF. Since the table is already in BCNF and no decomposition was done, all dependencies are preserved.

bank_account

1st FD...

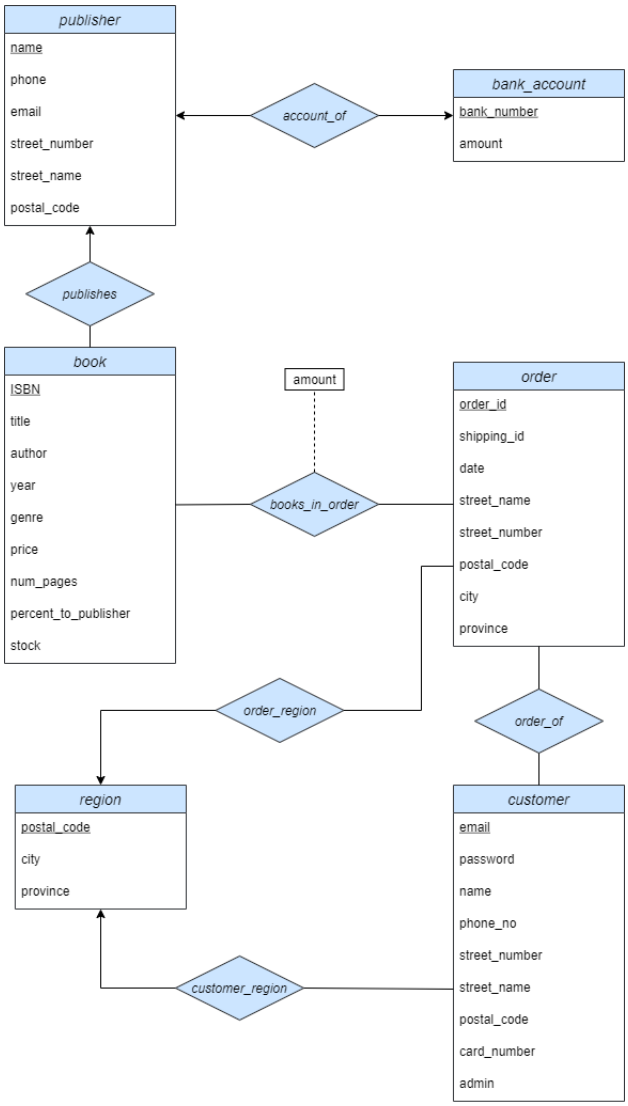
Closure of *bank_number*, (*bank_number*)+ = (bank_number, amount, debt_amount)

Since the closure of bank_number includes all attributes in the relation, it means bank_number is a superkey for the relation and it complies with BCNF.

Since all FD's for this relation satisfy the conditions for BCNF, this table is already in BCNF. Since the table is already in BCNF and no decomposition was done, all dependencies are preserved.

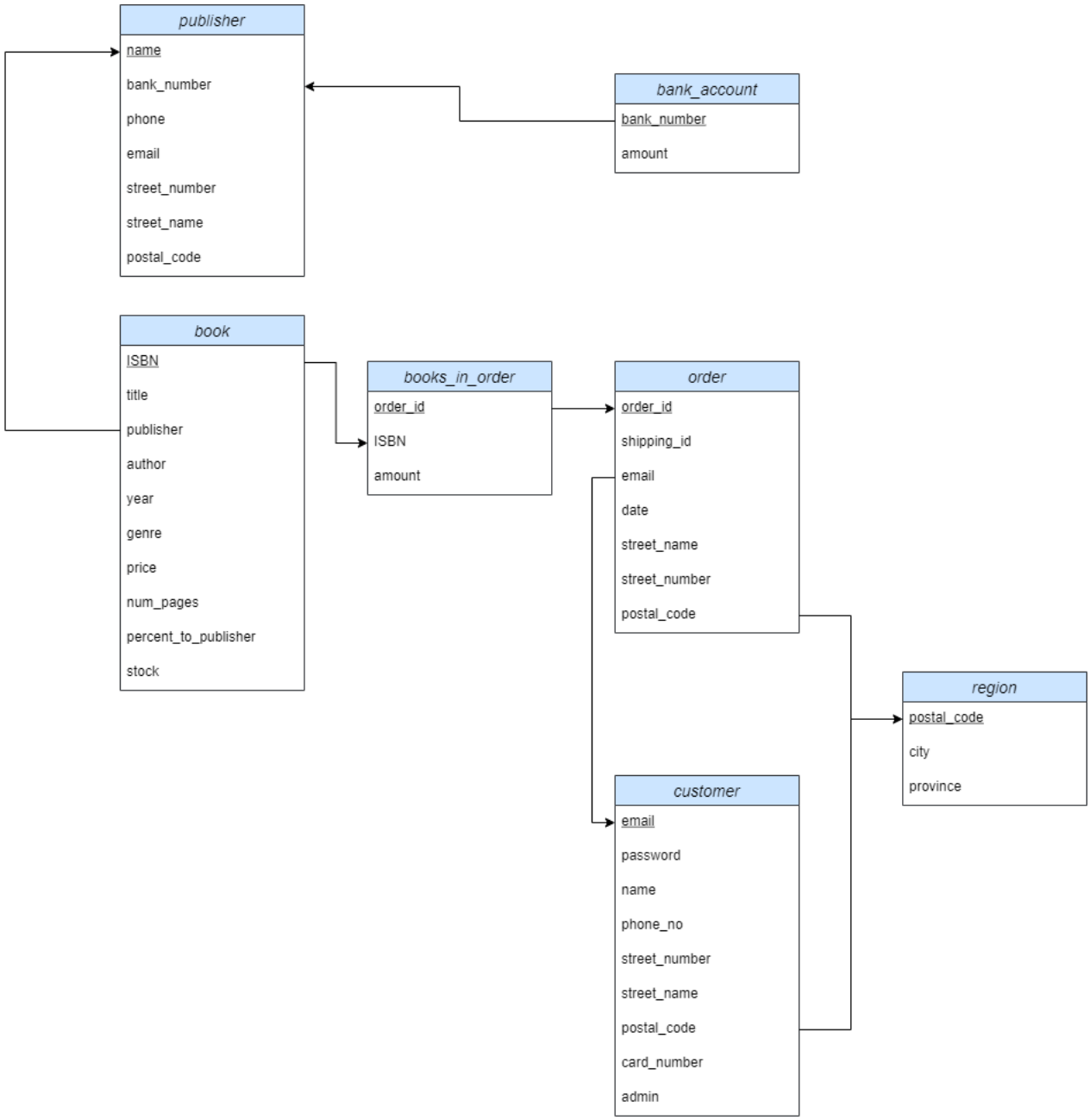
Results of Normalization

Since both the customer and order relations decomposed with a new relation called region_customer and region_order, we can merge the two together to create a new relation called region that both the customer and order relations will refer to. As such, this would be the updated ER diagram to show the new entities and relationships created by this new relation.



4 Database Schema Diagram

The following is the Schema Diagram created to model schemas gained from our ER diagram and after normalization.



5 Implementation

Implementation Design In this section we will briefly mention how we implemented certain features of our bookstore and what SQL elements were used to achieve this.

For our orders, the primary key that is used for the order relation is the order_id. The order_id is created immediately upon the creation of a new order into the table. We achieved this by making use of the "serial" datatype in our table.

Regarding the book stock, whenever the book's stock level goes below a certain threshold (ex. in our case we set it to below 10 books), an order needs to be placed to restock the books inventory. To achieve this, we implemented an SQL trigger on the book relation that activates whenever on stock of a book goes below 10. The trigger performs this check whenever a book entry is updated.

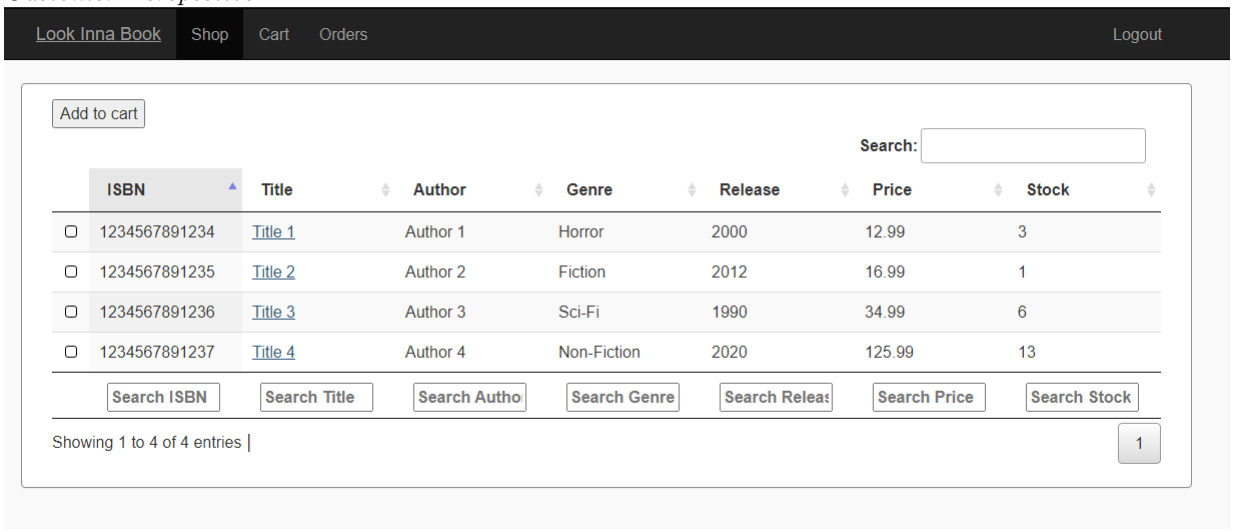
For publisher reports, we supported three different types of reports. Sales vs Expenditures (which is just

amount in bank_account vs debt_amount), Sales vs Authors and Sales vs Genres. For the last two reports, we implemented SQL functions that would return us a table with Authors/Genres and Sales for those given authors or genres. The function takes in a varchar argument that is the name of the publisher to retrieve the results for. Whenever a user requests to see the reports page for a specific publisher, we call those functions and pass in the specific publisher as the name. We also thought of using a view to do this, however we chose the route of SQL functions.

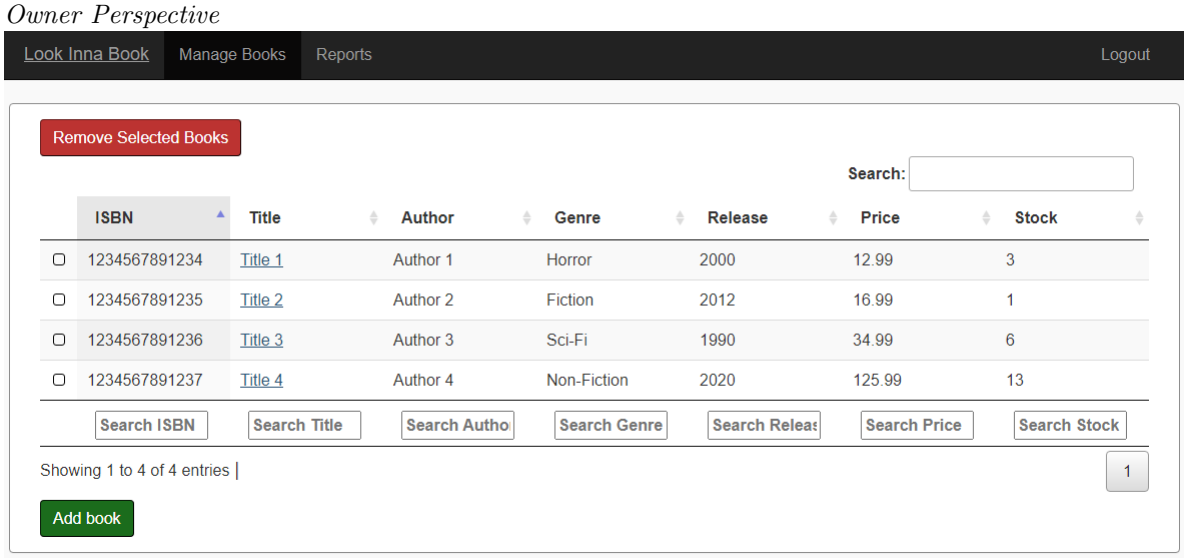
Implementation Tour

The following are a few screen shots showing what the user and owner would see when browsing the website.

Customer Perspective



Owner Perspective



Video Tour

The following is a video tour of the bookstore website. It walks through all the main features of the bookstore and demonstrates specific functionality mentioned in the problem statement.

View video here...

EDIT THIS!!!

6 Github Repository

All project files and source code can be found at the following Github repository...

7 Appendix I

Below are a list of three available time slots for December 18th 2021 (one day after the "implied" due date of Dec 17th 2021), for possible project demonstrations...

- 11:00AM - 11:20AM
- 12:00PM - 12:20PM
- 01:00PM - 01:20PM