Look-Inna-Book Report

Prepared by:
John Breton: 101129324
Ryan Godfrey: 101131511

Prepared for:
Ahmed El-Roby

COMP 3005
Carleton University

Winter 2020

Contents

1. CONCEPTUAL DESIGN

1.1 ER-Diagram

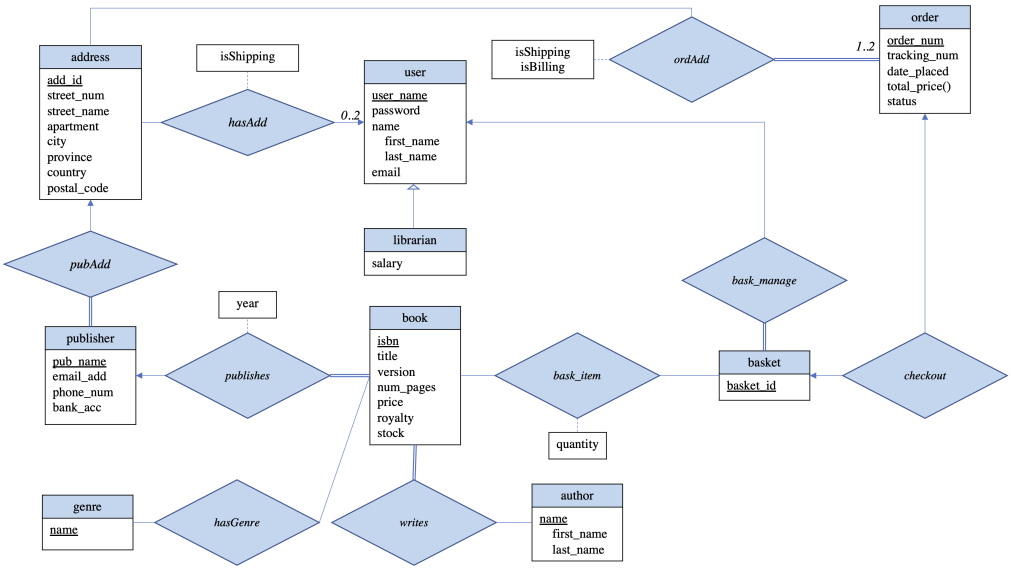

Figure 1.1: ER-Diagram

1.2 Entities

- **Book**
  - The "book" entity represents a book within the database. Its attributes include a title, version, number of pages, price, royalty, stock, and a unique ISBN. The ISBN is used as the primary key for this entity because no two books have the same ISBN number.
- **User**
  - The "user" entity represents a user of the bookstore. Its attributes include a username, password, name, and email. The name attribute is a multi-value attribute containing the user's first and last names. The username is used as the primary key for this entity because any two users are allowed to share passwords, names or emails.

- **Librarian**
  - The "librarian" entity is a specialization of the User entity. Its attributes are simply a username and a salary. This is because a librarian shares all attributes of a user, however, they also get paid and reserve the ability to modify the contents of the database. Username is the only primary key for this entity.
- **Address**
  - The "address" entity represents a physical street address. It has attributes such as street number, street name, apartment, city, province, country, and postal code. Due to the fact that entities can share the same address, a sequential address ID was included in the attributes to differentiate between instances.
- **Publisher**
  - The "publisher" entity represents a book publisher. It has attributes such as a name, email address, phone number and bank account number. The publisher's name is used as a primary key because it would be unreasonable to have multiple publishers with the same name.
- **Genre**
  - The "genre" entity represents a possible genre for the books. Its only attribute is the name of the genre (i.e. thriller, romance, children, etc.). Therefore, it's primary key is the name of the genre.
- **Author**
  - The "author" entity represents the authors of the books. Its only attribute is the author's name, which is a multi-valued attribute containing the first and last name of the author. Its primary key is the combination of the first and last name of the author because an author CAN share a name with another author, however there are no other attributes to distinguish the two authors, therefore it would be at the discretion of the user to determine which "John Smith" wrote the book.
- **Basket**
  - The "basket" entity represents a user's cart. This entity is designed to be related to multiple books in order to develop a selection of books for a user to purchase. It's only attribute is a sequential basket ID number that is also the primary key.

- **Order**
  - The "order" entity represents a basket that has been checked out and paid for. This entity is composed of attributes for the tracking number, date of order, total price of the order, status of the order and the unique order number. The primary key is the order number because no two tuples can share the same order number.

1.3 Relations

- **Writes**
  - The "writes" relation, relates an author to a specific book. This relation is intended to identify which authors wrote which books. The attributes of this relation are the primary keys of the author (name) and book (ISBN) entities. The primary key is the combination of these two attributes because any author can write multiple books and any book can have multiple authors.
- **Publishes**
  - The "publishes" relation relates a publisher to a specific book. This relation is intended to identify which publisher published which book. The attributes of this relation are the primary keys of publisher (name) and book (ISBN) as well as the year of publication. The primary key for this relation is the ISBN because a book cannot have more than one publisher.
- **hasGenre**
  - The "hasGenre" relation relates a genre to a specific book. This relation is intended to identify which genres a book is related to. The attributes of this relation are the primary keys of book (ISBN) and genre (name). The primary key for this relation is the combination of both attributes because a book cannot be related to the same genre twice.
- **bask_item**
  - The "bask_item" relation relates a book to a basket entity. This relation is intended to identify which books a user has chosen to "save" in their cart. The attributes of this relation are the primary keys of basket (basket ID), and book (ISBN) as well as a quantity attribute that denotes how many of that specific book the user would like to save. The primary key for this relation is the basket ID and ISBN because any book can be saved in multiple different baskets and a basket can contain multiple different books.

- **Checkout**
  - The "checkout" relation relates the basket entity to the order entity. This relation is intended to represent an "order completion" when a user decides to follow through with a purchase. It simply has the primary keys from basket and order as attributes (basket ID and order number); the primary key is the basket ID. This is because a basket instance cannot be "checked out" twice and an order number cannot relate to two different baskets, so it is a one-to-one relationship.
- **Bask_manage**
  - The "bask_manage" relation relates a user to a specific basket. This relation is intended to represent the relationship where a user "owns" their cart and can access it at any point. The attributes of this relation are basket ID and username and the primary key is basket ID because a user can create multiple baskets over time, however, two users cannot share a basket.
- **ordAdd**
  - The "ordAdd" relation relates an order to one or two specific addresses. This is intended to allow a user to create different addresses during the checkout process. The attributes of this relation are order number, the address ID and two boolean attributes to determine if it is a shipping, billing or both address. The primary key is the union of order number and the address ID because an address can be used for many orders and an order can have two addresses.
- **hasAdd**
  - The "hasAdd" relation relates a user to an address. This relation is intended to represent the existence of a user's addresses. Each "customer" will have exactly two addresses (one billing, one shipping) and each librarian does not have to be related to an address. The attributes for this relation are username, address ID and a boolean attribute to indicate if that address is the shipping address (if false, its the billing address). The primary key for this relation is address ID because a user can have multiple addresses (two) and but an address can only be used for 1 user.
- **pubAdd**
  - The "pubAdd" relation relates a publisher with an address. The relation is intended to represent the existence of a publisher's address. Each publisher must have one, and can only have one address. The attributes for this relation are publisher name and address ID with the primary key being the publisher name because of the limit to only one address.

1.4 Assumptions

1.4.1 Cardinalities

- **Address to User**
  - The cardinality for address to user is many-to-one, with the 'many' side being Address. This is because a user can have many addresses but an address can only be associated with one user. There is a cardinality constraint on "user" that limits the user to having a maximum of 2 addresses (1 shipping and 1 billing) the user could have 0 addresses if they were a librarian.
- **Basket to User**
  - The cardinality for basket to user is many-to-one with the 'many' side being Basket. This is because a user can have many different baskets (one at a time, but multiple over a period of time), however, a basket can only be associated with one user (no sharing).
- **Publisher to Address**
  - The cardinality for Publisher to Address is many-to-one with the 'many' side being Publisher. This is because a publisher can only have one address associated with it and, in theory, an address could be associated with multiple publishers (the implementation avoids this).
- **Book to Publisher**
  - The cardinality for Book to Publisher is many-to-one with the 'many' side being Book. This is because a publisher can publish multiple books, however, a book can only have one publisher (we assumed a book cannot be published twice or in conjunction with another publisher).
- **Book to Genre**
  - The cardinality for Book to Genre is many-to-many because a book can have multiple different genres and any genre could be used for any number of books.
- **Book to Author**
  - The cardinality foe Book to Author is many-to-many because a book can have multiple different authors and an author can write multiple different books.

- **Book to Basket**
  - The cardinality for Book to Basket is many-to-many because a book can be added to multiple different baskets and a basket can contain multiple books.
- **Order to Basket**
  - The cardinality for Order to Basket is one-to-one. This is because every order tuple is a result of one basket being "checked out" and every basket tuple has the ability to be checked out only once.
- **Order to Address**
  - The cardinality for Order to Address is many to many. This is because an address can be used for multiple different orders (in theory) and an order can be associated with multiple different addresses. There is a cardinality constraint on order that limits the number of associated addresses to be between 1 and 2. This is because an order must have at least one address (where the address is the shipping and billing address) or 2 addresses (separate shipping and billing).

1.4.2 Participation Types

- **Total Participation**
  - Book to Publishes is total participation because every book has to be published by someone (or else how is it in a bookstore).
  - Book to Writes is total participation because every book must have been written by at least one person.
  - Publisher to pubAdd is total participation because every publisher must have an address to ship things to.
  - Basket to bask_manage is total participation because every basket must be managed by a user, there shall not be a basket without an owner.
- **Patial Participation**
  - Every other relationship, that is not listed above, is partial participation because there is no constraint that requires the entity to participate in the relation.

2. REDUCTION TO RELATION SCHEMAS

- *book*(isbn, title, version, num_pages, price, royalty, stock)
- *user*(username, password, first_name, last_name, email)
- *librarian*(username, salary)
- *address*(add_id, street_num, street_name, apartment, city, province, country, postal_code)
- *publisher*(pub_name, email, phone_num, bank_acc)
- *genre*(name)
- *author*(first_name, last_name)
- *basket*(basket_id)
- *order*(order_num, tracking_num, date_placed, total_price, status)
- *writes*(first_name, last_name, isbn)
- *publishes*(isbn, pub_name, year)
- *hasGenre*(isbn, name)
- *bask_item*(basket_id, isbn, quantity)
- *checkout*(basket_id, order_num)
- *bask_manage*(basket_id, username)
- *ordAdd*(order_num, addid, isShipping, isBilling)
- *hasAdd*(add_id, user_name, isShipping)
- *pubAdd*(pub_name, add_id)

3. NORMALIZATION OF RELATION SCHEMAS

- *book*(isbn, title, version, num_pages, price, royalty, stock)

  ```
  F =
  {
  isbn -> title, version, num_pages, price, royalty, stock
  }
  ```

  $isbn^+ = \{isbn, title, version, num\_pages, price, royalty, stock\}$ therefore isbn is a superkey and the relation is in BCNF.

- *user*(username, password, first_name, last_name, email)

  ```
  F =
  {
  username -> password, first_name, last_name, email
  }
  ```

  $username^+ = \{username, password, first\_name, last\_name, email\}$ therefore username is a superkey and the relation is in BCNF.

- *librarian*(username, salary)

  ```
  F =
  {
  username -> salary
  }
  ```

  $username^+ = \{username, salary\}$ therefore username is a superkey and the relation is in BCNF

- *address*(add_id, street_num, street_name, apartment, city, province, country, postal_code)

  ```
  F =
  {
  add_id -> street_num, street_name, apartment, city, province,
            country, postal_code
  }
  ```

  $add\_id^+ = \{add\_id, street\_num, street\_name, apartment, city, province, country, postal\_code\}$ therefore add_id is a superkey and the relation is in BCNF

- *publisher*(pub_name, email, phone_num, bank_acc)

  ```
  F =
  {
  pub_name -> email, phone_num, bank_acc
  bank_acc -> pub_name
  email -> pub_name
  phone_num -> pub_name
  }
  ```

  $pub\_name^+ = \{pub\_name, email, phone\_num, bank\_acc\}$ therefore bank_acc, email, phone_num are all candidate keys so the relation is in BCNF.

- *genre*(name)

  ```
  F =
  {
  name -> name
  }
  ```

  name is trivial and, therefore, a superkey and the relation is in BCNF.

- *author*(first_name, last_name)

  ```
  F =
  {
  first_name, last_name -> first_name, last_name
  }
  ```

  (first_name, last_name) is trivial and, therefore, a superkey and the relation is in BCNF.

- *basket*(basket_id)

  ```
  F =
  {
  basket_id -> basket_id
  }
  ```

  basket_id is trivial and, therefore, it is a superkey and the relation is in BCNF.

- *order*(order_num, tracking_num, date_placed, total_price, status)

```
F =
{
order_num -> tracking_num, date_placed, total_price, status
tracking_num -> order_num
}
```

  $order\_number^+ = tracking\_num^+ = \{order\_num, tracking\_num, date\_placed, total\_price, status\}$ therefore order_num and tracking_num are both candidate keys and the relation is in BCNF.

- *writes*(first_name, last_name, isbn)

```
F =
{
first_name, last_name -> first_name, last_name
isbn->isbn
}
```

  $(first\_name, last\_name)^+ = \{first\_name, last\_name\}$ and $isbn^+ = \{isbn\}$; neither are superkeys, so in order to be in BCNF, (first_name, last_name) must be augmented with (isbn) to form a superkey (first_name, last_name, isbn) and it is now in BCNF.

- *publishes*(isbn, pub_name, year)

```
F =
{
isbn -> pub_name, year
}
```

  $isbn^+ = \{isbn, pub\_name, year\}$ therefore isbn is a superkey and the relation is in BCNF.

- *hasGenre*(isbn, name)

```
F =
{
isbn, name -> isbn, name
}
```

  (isbn, name) is trivial therefore (isbn, name) is a superkey and the relation is in BCNF.

- *bask_item*(basket_id, isbn, quantity)

  ```
  F =
  {
  basket_id, isbn -> quantity
  }
  ```

  $(basket\_id, isbn)^+ = \{basket\_id, isbn, quantity\}$ therefore (basket_id, isbn) is a superkey and the relation is in BCNF.

- *checkout*(basket_id, order_num)

  ```
  F =
  {
  basket_id -> order_num
  order_num -> basket_id
  }
  ```

  $basket\_id^+ = order\_num^+ = \{basket\_id, order\_num\}$ therefore order_num and basket_id are candidate keys and the relation is in BCNF.

- *bask_manage*(basket_id, username)

  ```
  F =
  {
  basket_id -> username
  }
  ```

  $basket\_id^+ = \{basket\_id, username\}$ therefore basket_id is a superkey and the relation is in BCNF.

- *ordAdd*(order_num, add_id, isShipping, isBilling)

  ```
  F =
  {
  order_number, add_id -> isShipping, isBilling
  }
  ```

  $order\_num^+ = add\_id^+ = \{order\_num, add\_id, isShipping, isBilling\}$ therefore (order_num, add_id) is a super key and the relation is in BCNF.

- *hasAdd*(add_id, username, isShipping)

  ```
  F =
  {
  add_id -> username, isShipping
  }
  ```

  $add\_id^+ = \{add\_id, username, isShipping\}$ therefore add_id is a superkey and the relation is in BCNF.

- *pubAdd*(pub_name, add_id)

  ```
  F =
  {
  pub_name -> add_id
  add_id -> pub_name
  }
  ```

  $pub\_name^+ = add\_id^+ = \{pub\_name, add\_id\}$ therefore pub_name and add_id are candidate keys and the relation is in BCNF.

4. DATABASE SCHEMA DIAGRAM

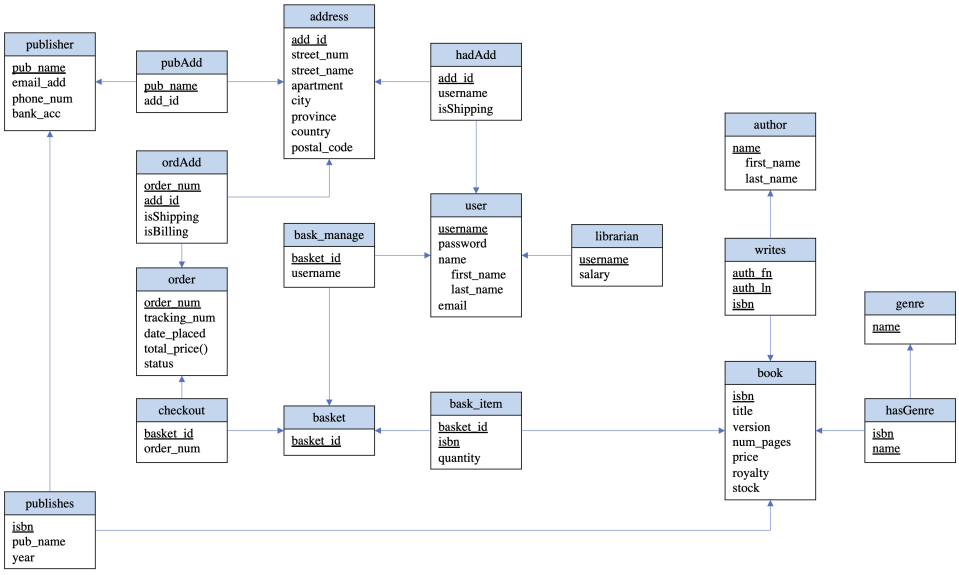

Figure 4.1: Database Schema Diagram

5. IMPLEMENTATION

5.1 Work-Flow

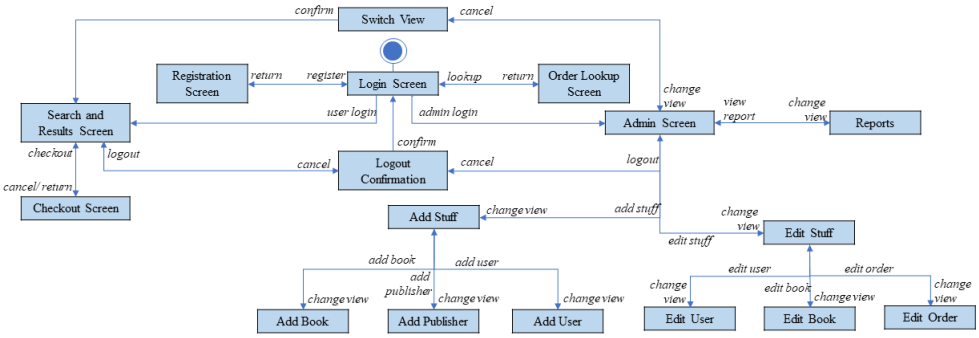

Figure 5.1: Work-Flow Diagram

5.2 Login Screen

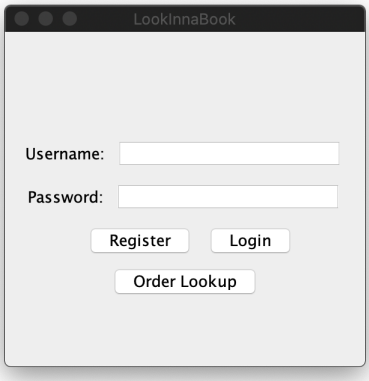

Figure 5.2: Login Screen

The login screen is the first screen that a user is presented with. Users enter their credentials in the required fields and press "Login" to access the bookstore. If a user enters valid customer credentials, they will transfer directly to the customer screen. If the user enters valid admin credentials, they will be prompted to choose which screen to transfer to (customer screen or admin screen). Users are also able to transfer to the registration screen or the order lookup screen by pressing the appropriate buttons.

5.3 Registration Screen

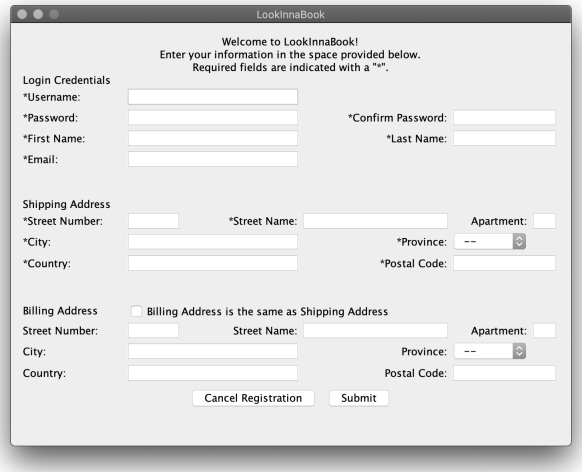

Figure 5.3: Registration Screen

The registration screen provides the ability for a user to register as a new customer. The user can cancel the registration and return to the login screen, or fill out the required fields with valid information and submit their registration. Once successfully submitting their information, the user will be prompted to return to the login screen to login.

5.4 Order Lookup Screen

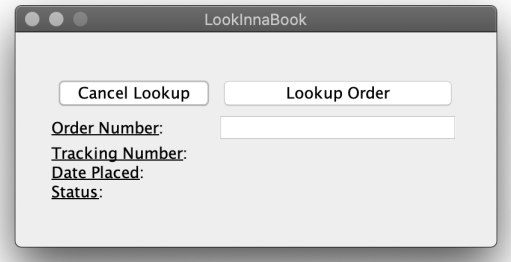

Figure 5.4: Order Lookup Screen

The order lookup screen provides the ability for a user to lookup an existing order by searching the order number. The user can return to the login screen by pressing “cancel lookup” or they can get the order information by entering the order number and pressing “lookup order”.

5.5 User Screen

The user screen is the screen that allows users of the bookstore to search and purchase books.

5.5.1 Customer Screen

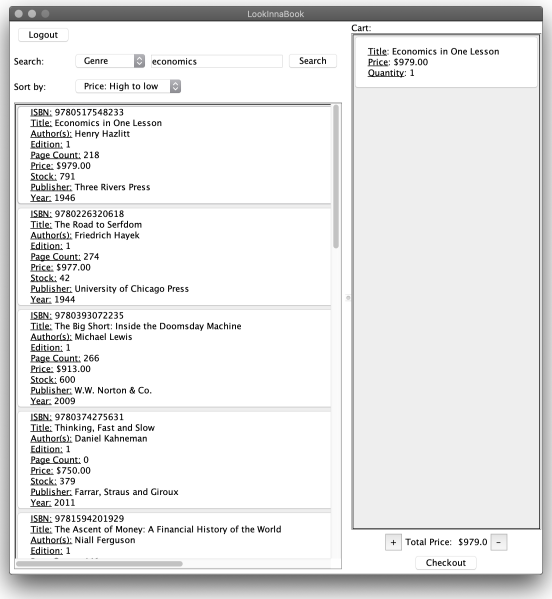

Figure 5.5: Customer View Screen

The customer screen allows users to search books by title, genre, author, ISBN, publisher or year by selecting the appropriate search filter, entering the required search term and pressing “search”. Once results are displayed, a user can sort the results by price, year or title. The user can add a book to their cart by clicking the appropriate book. The user can increase or decrease the quantity of a book in their cart by selecting the book in their cart and selecting “+” or “-” to add and subtract respectively. Once the user is satisfied with their cart, they can proceed to the checkout screen by pressing “checkout”. The user is also able to logout at any time by pressing the appropriate button.

5.5.2 Checkout Screen

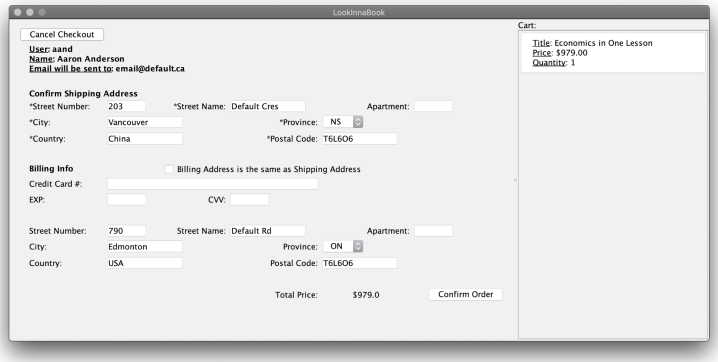

Figure 5.6: Checkout Screen

The checkout screen allows a user to complete an order and purchase the books that were in their cart. The user's information is displayed at the top of the screen and the address fields are populated with the information that was provided during registration. The user can view their cart (that they're buying) on the right. A user can edit their address information, as well, they must provide credit card information to complete the transaction. A user can cancel the order by selection "cancel checkout" or complete the order by selecting "confirm order". If a user cancels their order, their cart will remain the same, if a user completes the order, their cart will be emptied.

5.6 Admin Screen

The admin screen allows users that work for the bookstore, to add entities and edit properties of entities within the bookstore. They are also able to view reports. Users can log out of the bookstore or switch to the customer view at any time by using the drop down menu located in the upper right of the screen.

5.6.1 Add Stuff

The add stuff tab allows users to add entities to the database.

Add Book

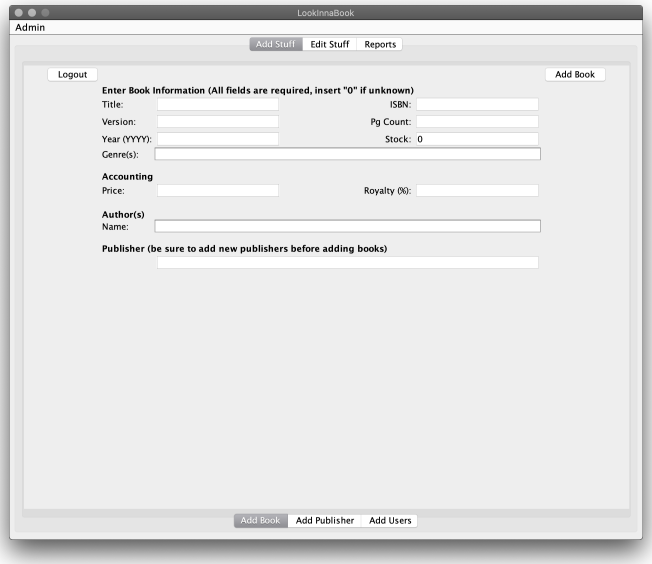

Figure 5.7: Add Book Screen

The add book screen is the first screen that admin users are presented with. This screen allows users to add a book, that doesn't already exist, to the database. After users enter valid information, they can submit the book by pressing "add book". Users can switch to the "add publisher", "add user", "edit stuff" or "reports" screens by selecting the appropriate tabs.

Add Publisher

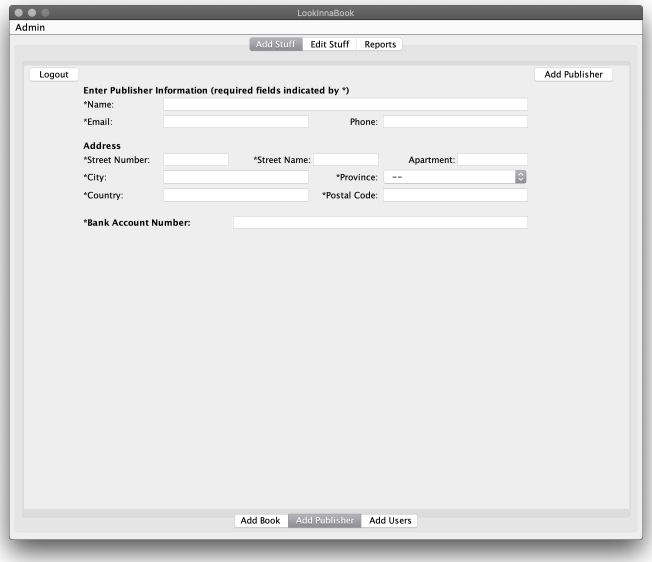

Figure 5.8: Add Publisher Screen

The add publisher screen allows an admin to add a publisher to the database. The admin can add a publisher by entering valid information and pressing the "add publisher" button. The user can logout at any time by pressing the "logout" button. The user can also switch to the "add book", "add user", "edit stuff" or "reports" screens by selecting the appropriate tabs.

Add User

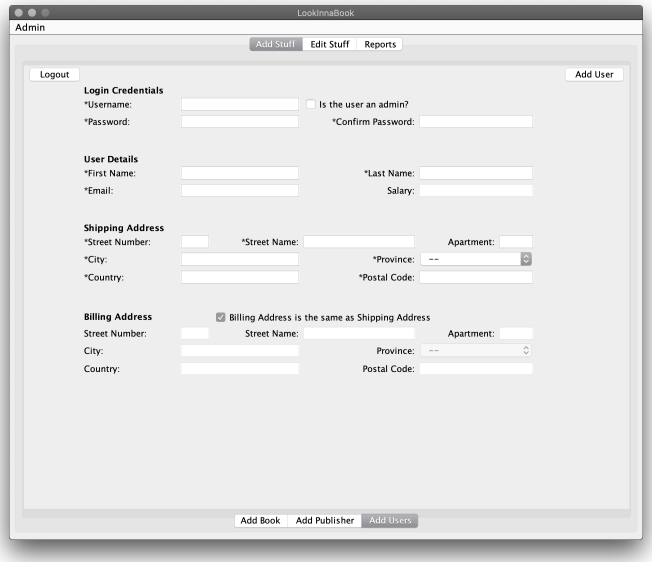

Figure 5.9: Add User Screen

The add users screen allows an admin to add a user to the database. The admin can add a user by entering valid information and pressing the “add user” button. The user can logout at any time by pressing the “logout” button. The user can also switch to the “add book”, “add publisher”, “edit stuff” or “reports” screens by selecting the appropriate tabs.

5.6.2 Edit Stuff

The edit stuff screen allows admin users to edit existing entities.

Edit Book

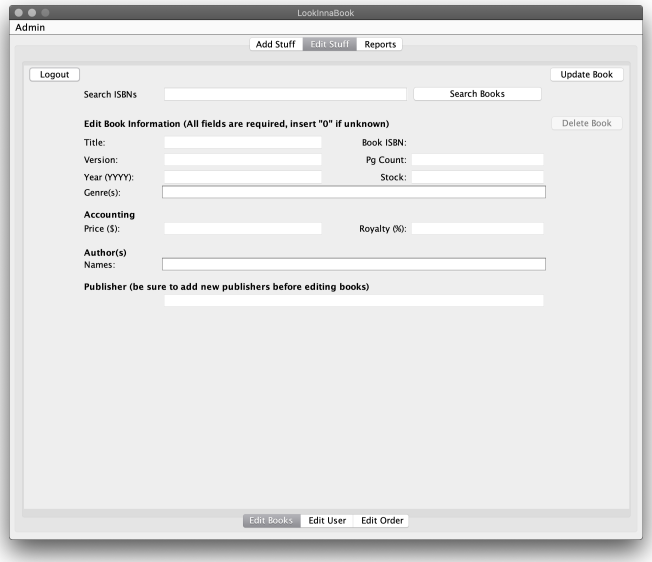

Figure 5.10: Edit Book Screen

The edit book screen allows users to edit a book that already exists in the database. After the user enters a valid isbn in the search field and presses “search books”, the fields will populate with the book’s current information. The user is then able to edit the information and submit the edited information by pressing “update book”. Users can switch to the “add stuff”, “edit user”, “edit order” or “reports” screens by selecting the appropriate tabs. The user can log out at any time by pressing the “logout” button.

Edit User

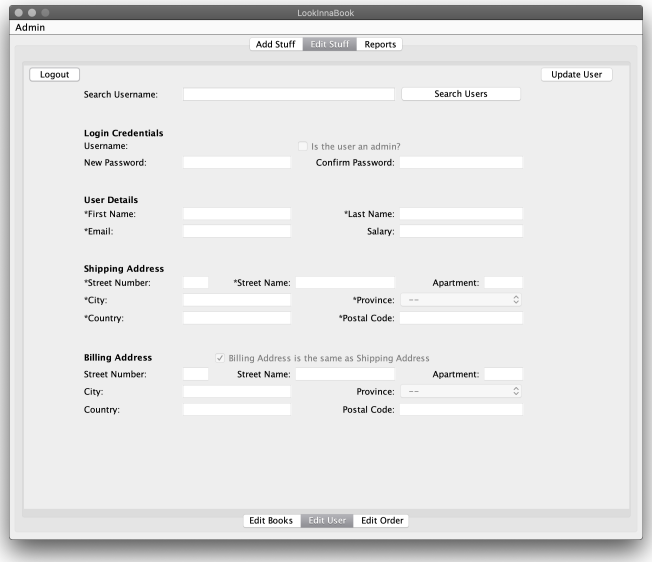

Figure 5.11: Edit User Screen

The edit user screen allows users to edit a user profile that already exists in the database. After the user enters a valid username in the search field and presses "search users", the fields will populate with the user's current information. The admin is then able to edit the information and submit the edited information by pressing "update user". Admins are not able to edit their own information. Users can switch to the "add stuff", "edit books", "edit order" or "reports" screens by selecting the appropriate tabs. The user can log out at any time by pressing the "logout" button.

Edit Order

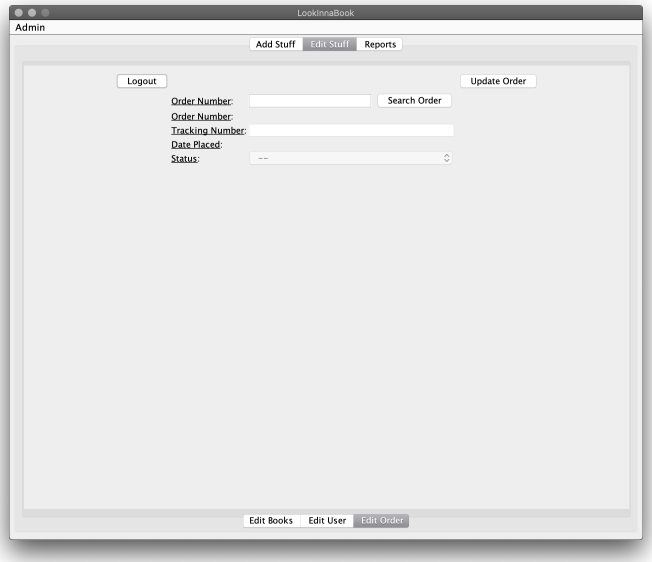

Figure 5.12: Edit Order Screen

The edit order screen allows admins to edit an order that already exists in the database. After the user enters a valid order number in the search field and presses "search order", the fields will populate with the order's current information. The user is then able to edit the information and submit the edited information by pressing "update order". Users can switch to the "add stuff", "edit books", "edit user" or "reports" screens by selecting the appropriate tabs. The user can log out at any time by pressing the "logout" button.

5.6.3 Reports

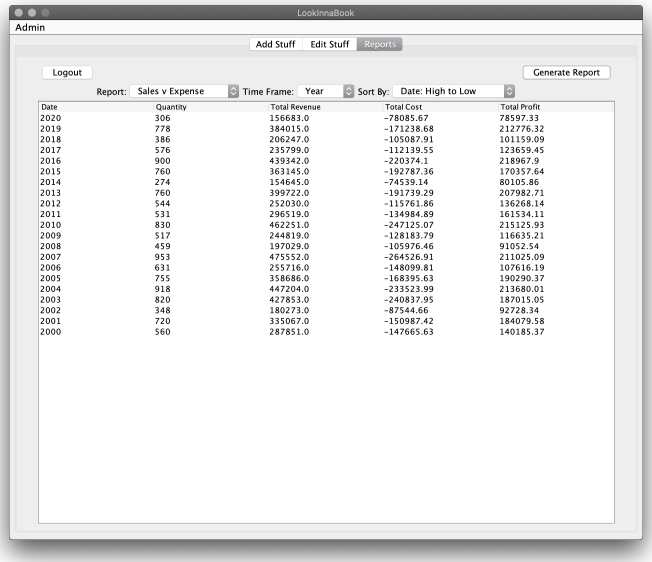

Figure 5.13: Reports Screen

The reports tab allows admins to generate and view information about sales over a period of time. The user can select which report to generate ("Sales v Expense", "Sales per genre", "Sales per author", "Sales per publisher"), the time frame for the report and how to sort the report. Once all valid options are selected, the user presses "generate report" to display the report. For all reports except "Sales v Expense", the time frame option refers to the data in the past time period (i.e. past year). In the "Sales v Expense" report, the time frame option determines how to split the data (i.e. by year). Users can switch to the "add stuff" or "edit stuff" screens by selecting the appropriate tabs. The user can log out at any time by pressing the "logout" button.

5.7 Switch Screens

When users request to switch screens, they are prompted with options to confirm their intended actions.

5.7.1 Confirm Logout

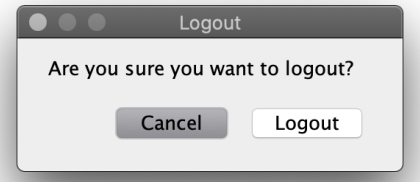

Figure 5.14: Confirm Logout Screen

The confirm logout screen is shown whenever a user requests to logout. The user can log out by pressing logout, or dismiss the screen by pressing cancel.

5.7.2 View Switch

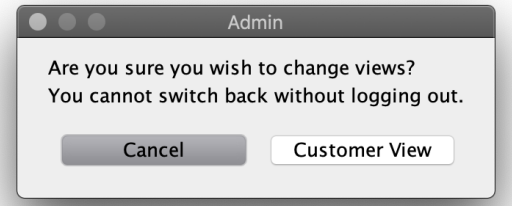

Figure 5.15: Confirm View Switch Screen

The view switch screen is shown when an admin requests to switch to the user view from their admin view. The user can continue to the user screen by selecting "customer view" or they can dismiss the window and return to the admin view by selecting "cancel".

5.7.3 Admin Login Option

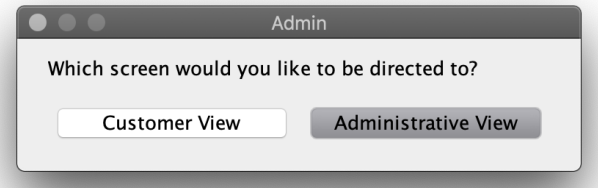

Figure 5.16: Screen Choice Screen

The admin login option screen is shown when an admin logs in. This window allows the user to decide whether to proceed to the customer view or the admin view.

5.8 Scenarios

5.8.1 New User

1. The user opens the application and is presented with the login screen.
2. The user hits the "Register" button and is taken to the registration screen.
3. The user proceeds to enter their information into the appropriate fields. Once finished, they submit their application and are then informed they can now login.
4. The user returns to the login page via a button press and enters their username and password. They hit the "Login" button, at which point their account is authenticated and they are successfully brought to the store page.
5. The user looks for their favourite book "Animal Farm" (they are weird). Upon entering this into the search box, they hit search and one result is returned. Happy that their book is available, they click the book and it is added to their cart.
6. "I could use 7 copies", thinks the user. As such, they select the book in their cart and hit the "+" button until they have 7 copies of Animal Farm in their cart.
7. "You know, I could go for every book written in 2004. What a great year! No COVID-19, no worries about coops during the summer. Just good old 2004", thinks the user. They change the search type to be by year, they enter 2004, and then they hit search. They are happy to have so many results return.

8. Feeling a bit overwhelmed, the user sorts the results of the search by price, from low to high, and selects the 10 cheapest books written in 2004.

9. The user proceeds to the checkout page via the "Checkout" button. They enter their credit card information that is absolutely not stored in the database at any point. They receive confirmation that their order was successful! The user notes the order number.

10. The user returns to the store page and proceeds to logout via the "Logout" button. They confirm the logout, and are presented with the login page. Feeling satisfied, they exit the application.

5.8.2 Admin

1. An admin opens the application and they are presented with the login screen. They enter their credentials. Upon authentication, they are prompted as to whether they wish to go to the customer view or the administration view. They select the administration view.

2. They navigate to the edit publisher tab, and update the bank account number for the publisher "Penguin Books". After the update is submitted, the admin remembers that they just hired a new employee and that they needed to be added to the system.

3. Navigating to the add user tab, the admin inputs all of the employee information. Stupidly, they forgot to mark that the newly inserted user was an admin.

4. The admin navigates to the edit user tab, and they update the new employee's information to indicate that they are an admin.

5. Finally, the admin navigates to the report tab. The admin creates a report for the sales by author for the past day. "Why the hell is George Orwell so popular? Who the hell bought 7 of his books today?". Perplexed, the admin continues to analyze the report. They prepare the order of the books for the most recent order and set the status as "On the way!", being sure to input the tracking number for the postage label they just purchased for exactly $16.72 from Canada Post. Feeling satisfied, the admin confirms that they wish to logout.

5.8.3 Order Lookup

1. "I wonder how my order is doing", thinks the user. They open the application and are presented with the login page. They navigate to the order lookup page via a button press on the login page.

2. The user enters their order number and notices that their order is “On the way!”. “How exciting!”, the user thinks. Satisfied, the user exits the application.

5.8.4 Returning User

1. A user opens the application and enters their login information. Upon being authenticated, they are brought to the store page.

2. “Oh thank goodness my 27 copies of The Silmarillion are still in my cart. Maybe I’ll buy those some day”, thinks the user. Happy, they confirm their logout and exit the application.

6. Bonus Features

6.1 Graphical User Interface

Although we were tasked with creating a program, there was no specification saying the application had to have a GUI. Our GUI is easy to use, containing tool-tips for all possible interactions may have with the application. Additionally, some quality of life options such as seeing the cart during checkout, the ability to sort results, and feedback if an incorrect input is entered all augment the experience for the user. This in turn elevates our application to be not only functionally compliant of the project requirements, but indeed a usable implementation of said requirements.

6.2 Order Lookup

Although the project does specify that a user can lookup an order, our implementation allows the user additional useful information, such as an orders tracking number, the date the order was placed, and the current status of the order.

6.3 Ability to Add/Remove Books

The project does not specify any need for admins to add or remove books. In our application, admins do have this ability, along with the ability to add publishers, and a full range of editing options for items within the database.

6.4 Ability to Edit Orders

The project does not mention anything about editing orders once they have been placed. Despite this, admins have the ability to lookup an order via an order number and can change the status of the order. They can also change the tracking number should that need to be updated.

6.5 Approximate Searching

Users have the ability to enter incomplete search text into the search bar. The system will attempt to match the results to any books that contain the pattern entered by the user. For example, the user can search by title with simply the word "Lord". The system will return all books that contain the word "Lord" at any position within the book title.

7. GITHUB REPOSITORY

The GitHub repository for the project can be found using the following link: https://github.com/john-breton/COMP3005