BOOKSTORE DATABASE MANAGEMENT SYSTEM REPORT

Look Inna Book Inc.

Author: Addison Xiong

Course: COMP 3005 2022F

Table of Contents

1. Problem Statement

Design and implement an application for an online bookstore (Look Inna Book). This application lets users browse a collection of books that are available in the bookstore. A user can search the bookstore by book name, author name, *ISBN*, genre, etc. When a book is selected, information on the author(s), genre(s), publisher, number of pages, price, etc. can be viewed. A user can select as many books as she likes to be added to the checkout basket. A user needs to be registered in the bookstore to be able to checkout. When checking out, the user inserts billing and shipping information (can be different than those used in registration) and completes the order. The bookstore has the feature of tracking an order via an *order number*. A user can use this order number to track where the order is currently. Although shipping is carried out by a third-party shipping service, the online bookstore should have the tracking information available for when the user inquires about an order using the order number. Assume all books are shipped from only one warehouse (no multiple order numbers for multiple books shipped from multiple warehouses). The bookstore owners can add new books to their collections or remove books from their store. They also need to store information on the publishers of books such as name, address, email address, phone number(s), banking account, etc. The banking account for publishers is used to transfer a percentage of the sales of books published by these publishers. This percentage is variable and changes from one book to another. The owners should have access to reports that show sales vs. expenditures, sales per genres, sales per author, etc. The application should also be able to automatically place orders for new books if the remaining quantity is less than a given threshold (e.g., 10 books). This is done by sending an email to the publisher of the limited books to order several books equal to how many books were sold in the previous month(not required).

2. Conceptual Design

Before designing the ER-diagram and Relation Schemas, it should be clearer to first make appropriate assumptions, list the entities and summarize the relationships based on the problem statement above.

2.1 Entities

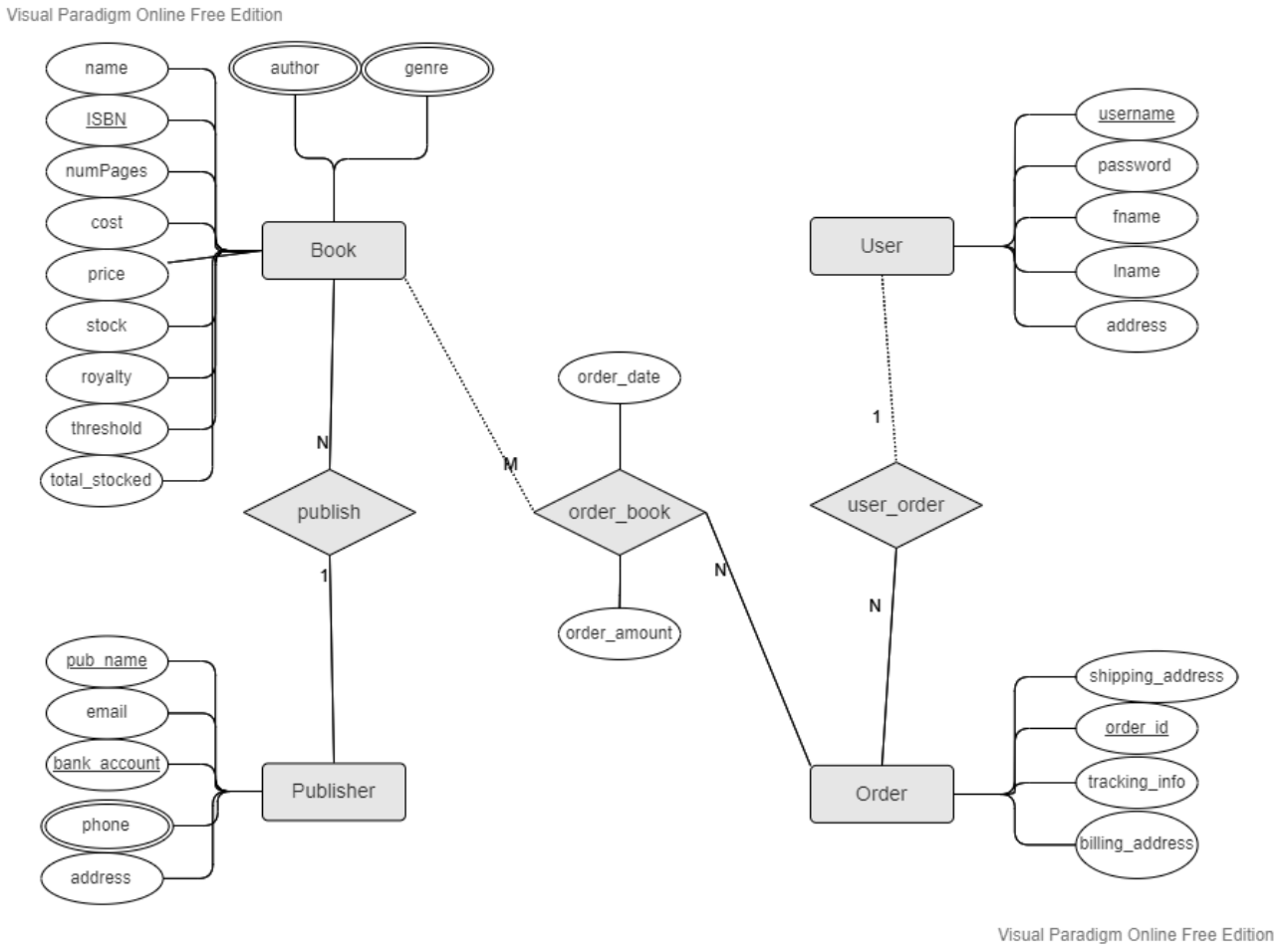
- Book: name, author(s), ISBN, genre(s), numPages, cost, price, stock, royalty, threshold, total_stocked
- User: username, password, fname, lname, address
- Order: order_id, shipping_address, billing_address, tracking_info
- Publisher: pub_name, address, email, phone(s), bank_account

2.2 Relationships and Assumptions

- Publish: A book can only have 1 publisher, and a publisher can publish 1 to N books
    - Publisher (1, total participation): Book (N, total participation)
        - ⇨ No need for a new table, add publisher_fk to Book table
- User_Order: A user can place 0 to N orders, and each order is associated with 1 user
    - User (1, partial participation): Order (N, total participation)
        - ⇨ No need for a new table, add user_fk to Order table
- Order_Book: an order can have 1 to N books, each book can be associated with 0 to M orders
    - Order (M, total participation): Book (N, partial participation)
        - ⇨ Need a new table, also add an order_amount for each book and order_date

Now, it's clear to draw an ER Diagram based on the above entities, relationships, and assumptions.

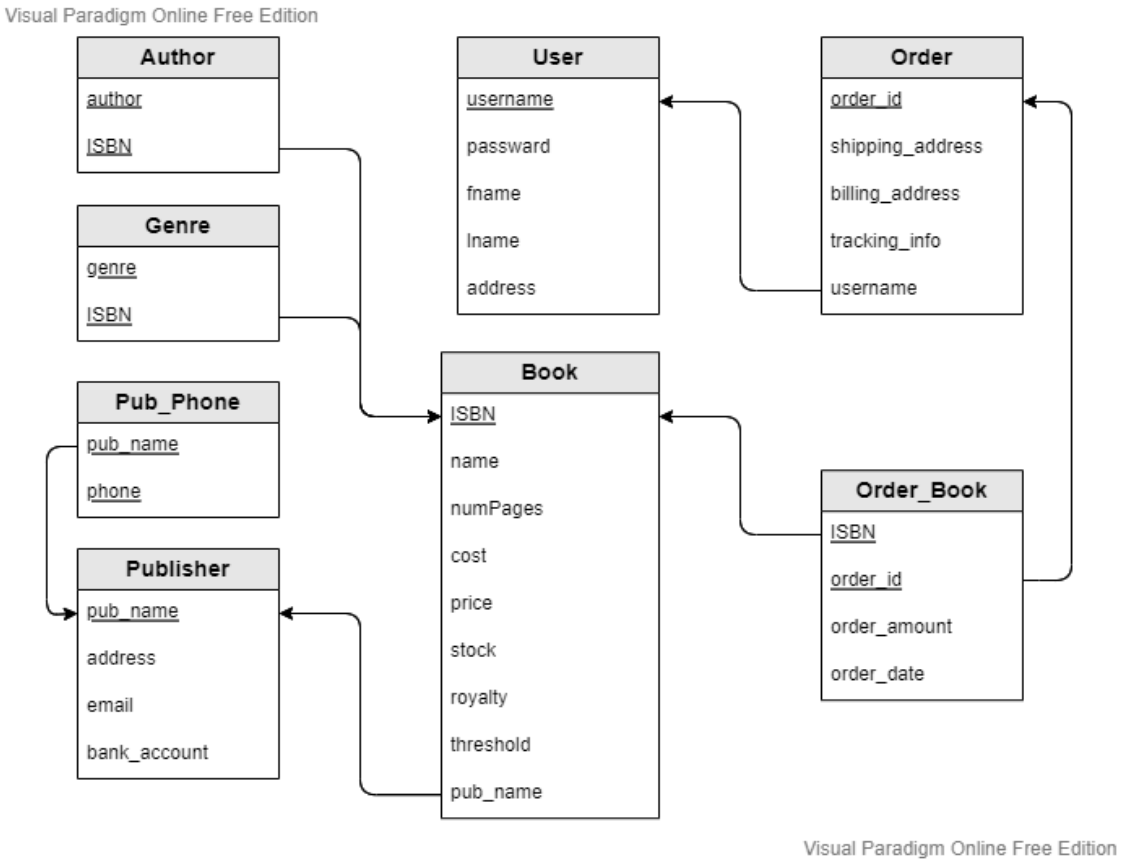
2.3 ER Diagram (Chen Notation)



2.4 Reduction to Relation Schemas

- Book (<u>ISBN</u>, name, numPages, cost, price, stock, royalty, threshold, pub_name)
- Author (<u>author</u>, <u>ISBN</u>)
- Genre (<u>genre</u>, <u>ISBN</u>)
- User (<u>username</u>, password, fname, lname, address)
- Publisher (<u>pub_name</u>, address, email, <u>bank_account</u>)
- Pub_Phone (<u>pub_name</u>, <u>phone</u>)
- Order (<u>order_id</u>, shipping_address, billing_address, tracking_info, username)
- Order_Book: (<u>ISBN</u>, <u>order_id</u>, order_amount, order_date)

Relation Schema diagram v1 before normalization test is also shown in the next page.



3. Normalization of Relation Schemas

3.1 Normalization Test for Each Relation

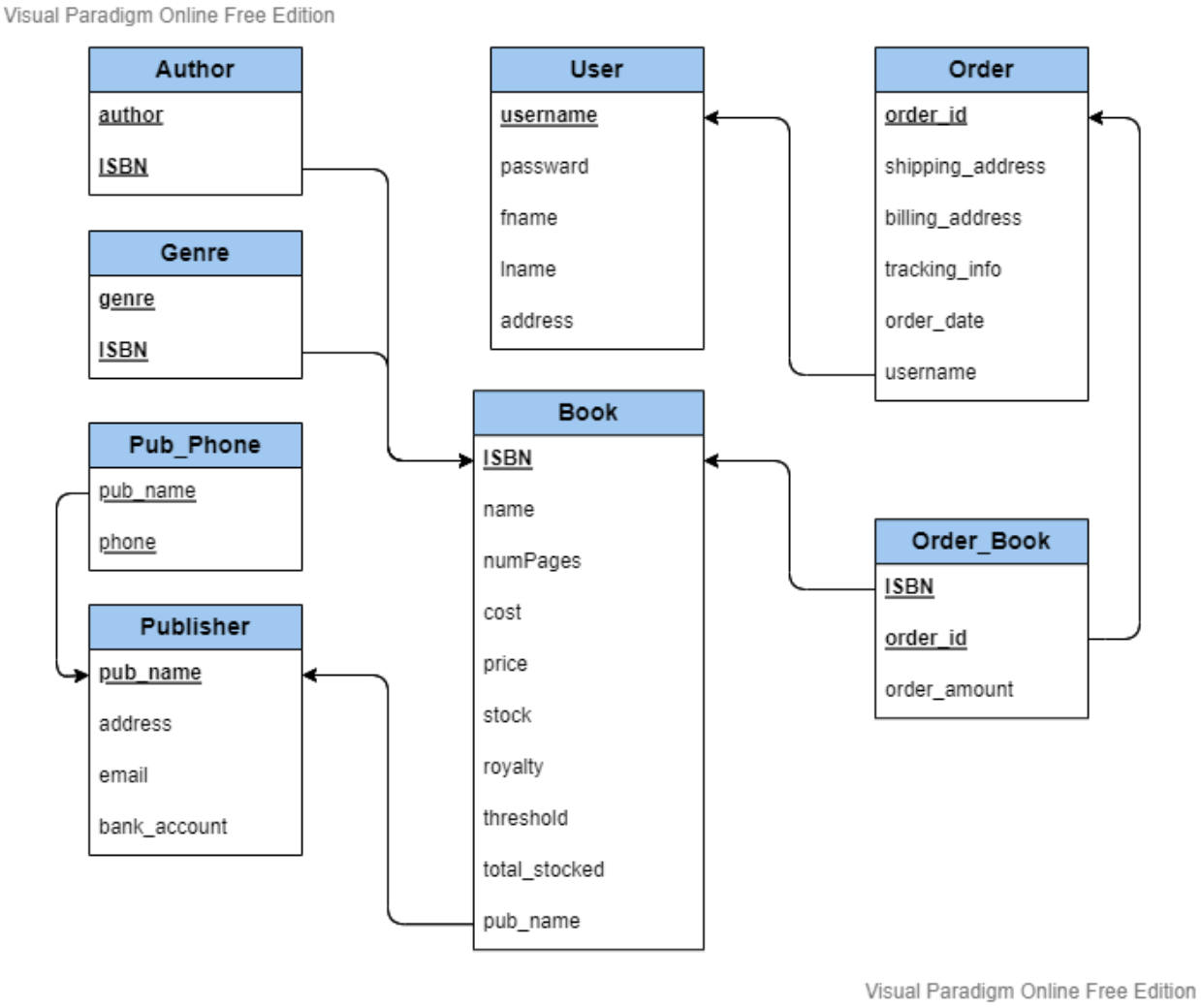
- Book (ISBN, name, numPages, cost, price, stock, royalty, threshold, total_stocked, pub_name)
  - F =
    {
    ISBN -> name, numPages, cost, price, stock, royalty, threshold, total_stocked, pub_name
    }
  - (ISBN ∪ RHS) = R
    - ⇨ ISBN is a superkey
  - Function dependencies that are derived from augmentation rules and decomposition rules all have LHS as a superkey, and all other function dependencies are all trivial.
    - ⇨ this relation is in BCNF
- Author (author, ISBN)
  - F =
    {
    author, ISBN -> ISBN
    author, ISBN -> author
    }

- All function dependencies are trivial, and LHS is a superkey
  - ⇨ This relation is in BCNF

- Genre (genre, ISBN)
  - F =
    {
     genre, ISBN -> ISBN
     genre, ISBN -> genre
    }
  - All function dependencies are trivial, and LHS is a superkey
    - ⇨ This relation is in BCNF
- User (username, password, fname, lname, address)
  - Multiple users are allowed to register on the same address, so the address can not determine the user.
  - F =
    {
     username -> password, fname, lname, address
    }
  - (username ∪ password, fname, lname, address) = R
    - ⇨ username is a superkey
  - Function dependencies that are derived from augmentation rules and decomposition rules all have LHS as a superkey, and all other function dependencies are all trivial.
    - ⇨ this relation is in BCNF
- Publisher (pub_name, address, email, bank_account)
  - Different businesses can share an address or an email, but they should have different bank account. That is to say, the bank account uniquely determines publisher.
  - F =
    {
     pub_name -> address, email, bank_account
     bank_account -> pub_name
    }
  - Other function dependencies are all trivial
  - $bank\_account^+$ = {pub_name, address, email, bank_account}
    - ⇨ pub_name and bank_account are superkeys
  - Function dependencies that are derived from augmentation rules and decomposition rules all have LHS as a superkey, and all other function dependencies are all trivial.
    - ⇨ this relation is in BCNF
- Pub_Phone (pub_name, phone)
  - F =
    {
     pub_name, phone -> pub_name
     pub_name, phone -> phone
    }
  - All function dependencies are trivial, and LHS is a superkey
    - ⇨ This relation is in BCNF

- Order (order_id, shipping_address, billing_address, tracking_info, username)
    - Since shipping address and billing address can be different than the address used for registration, username can not determine the address used in an order.
    - F =
      {
      order_id -> shipping_address, billing_address, tracking_info, username
      }
    - (order_id ∪ shipping_address, billing_address, tracking_info, username) = R
        - ⇨ order_id is a superkey
    - Function dependencies that are derived from augmentation rules and decomposition rules all have LHS as a superkey, and all other function dependencies are all trivial.
        - ⇨ this relation is in BCNF
- Order_Book: (ISBN, order_id, order_amount, order_date)
    - F =
      {
      ISBN, order_id -> order_amount, order_date
      order_id -> order_date
      }
    - $order\_id^+$ = { order_id, order_date}
        - ⇨ order_id is not a superkey, this relation is NOT in BCNF
        - ⇨ BCNF Decomposition:
            - OrderDate(order_id, order_date)
            - Order_Book(ISBN, order_id, order_amount)
- Order_Book: (ISBN, order_id, order_amount)
    - F =
      {
      ISBN, order_id -> order_amount
      }
    - (ISBN, order_id ∪order_amount) = R => LHS is a superkey
    - Function dependencies that are derived from augmentation rules and decomposition rules all have LHS as a superkey, and all other function dependencies are all trivial.
        - ⇨ this relation is in BCNF
- OrderDate(order_id, order_date)
    - This relation has composite relationship with Order because its PK is pointing to Order's PK, and it's considered as a part of the Order. Every time the records are deleted/added when the correspondences are deleted/added in Order table.
    - Need to composite this relation into Order relation, and run BCNF test again on Order
- Order (order_id, shipping_address, billing_address, tracking_info, username, order_date)
    - F = {
      order_id -> shipping_address, billing_address, tracking_info, username }
        - ⇨ order_id is a superkey
    - Function dependencies that are derived from augmentation rules and decomposition rules all have LHS as a superkey, and all other function dependencies are all trivial.
        - ⇨ this relation is in BCNF

3.2 Database Schema Diagram After Normalization



4. Implementation

4.1 Languages, frameworks, and software

- Programming languages: C++, SQL
- Frameworks and Tools: QT(5.12.8 GCC 9.3.0 64 bit), QT Creator(4.11.0), QT Designer, QT SQL, SQLite(3.31.1)

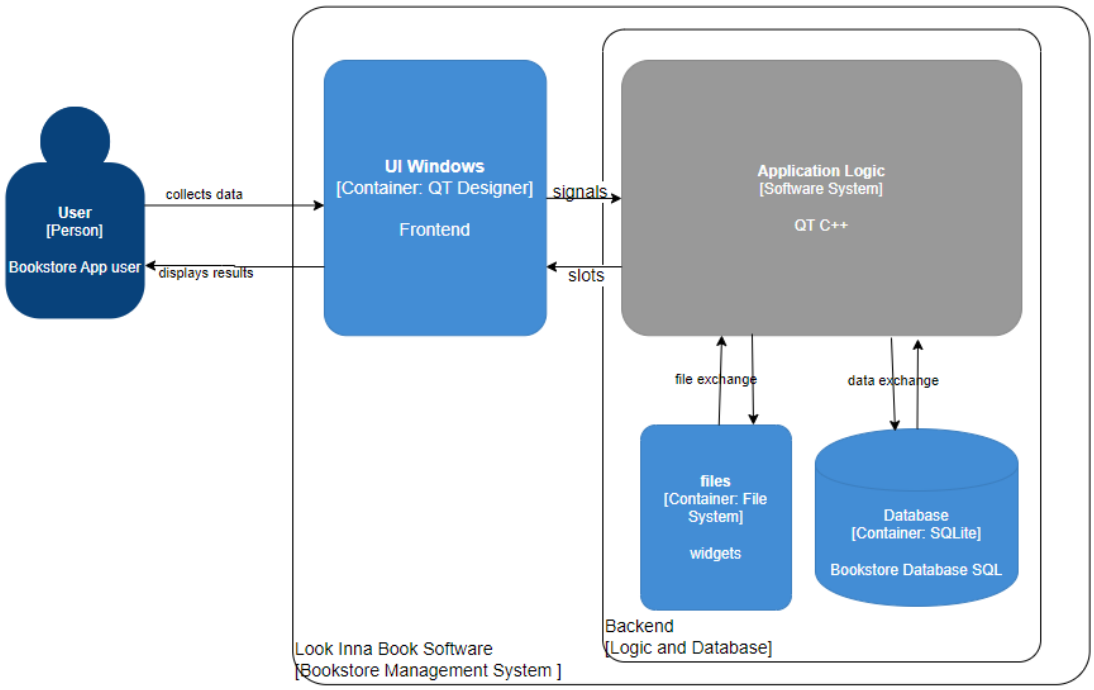
Note that the QT SQL module uses driver plugins to communicate with the different database APIs. SQLite version 3 is included as a third-party library within Qt, you can connect to SQLite database engine without hosting a database server. SQLite operates on a single file, which must be set as the database name when opening a connection. If the file does not exist, SQLite will try to create it.

Since we don't need to host a database server, it makes SQLite a great choice for the purpose of making this bookstore management system very lightweight and easy to test. However, please be aware that the datatypes in SQLite are different than other traditional SQL databases management systems. The

datatypes used in this project is compatible with SQLite only. Each value stored in an SQLite database has one of the following storage classes:

1. NULL. The value is a NULL value.
2. INTEGER. The value is a signed integer, stored in 0, 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
3. REAL. The value is a floating-point value, stored as an 8-byte IEEE floating point number.
4. TEXT. The value is a text string, stored using the database encoding (varchar, char, etc.).
5. BLOB. The value is a blob of data, stored exactly as it was input.

4.2 Application Architecture

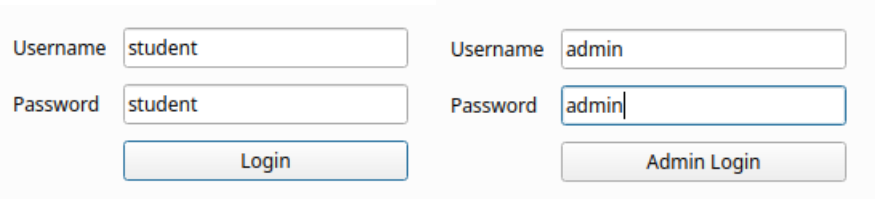


- UI Windows:
    - Forms(xml):
        - mainwindow.ui
            - login, admin login, register
            - browse/search/sort books, place order, track order
        - adminwindow.ui
            - add/remove books
            - generate report

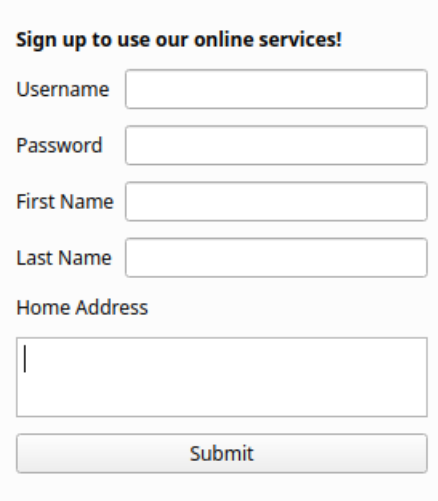
- Application Logic:
    - MainWindow class (control/boundary): has AdminWindow class
        - Establishes connection with the database
        - Initializes database tables and generates dummy data
        - Handles all user's requests (UI events)
        - Exchanges data with the database
        - Opens AdminWindow on request
    - AdminWindow class (control/boundary):
        - Establishes connection with the database
        - Handles admin's requests (UI events)
        - Exchanges data with the database
- Database:
    - In addition to the eight tables that are shown in the last section, there are one sqlite_sequence system table, 2 views and 3 triggers:
        - sqlite_sequence: system table to keep track of autoincrement primary key for the order table
        - table_view: list all books with genres and authors
        - report_view: list price and sales info group by books
        - restock_book: restocks 30 books if the stock is lower than the threshold
        - init_total_stocked: initialize total_stocked when insert new book
        - update_total_stocked: update total_stocked when the bookstore makes a purchase to stock more books
    - Dummy data are inserted at the start of the application.
        - You can login as a user using
            - Username: student
            - Password: student
        - You can login as an administrator using
            - Username: admin
            - Password: admin

4.3 User Screens

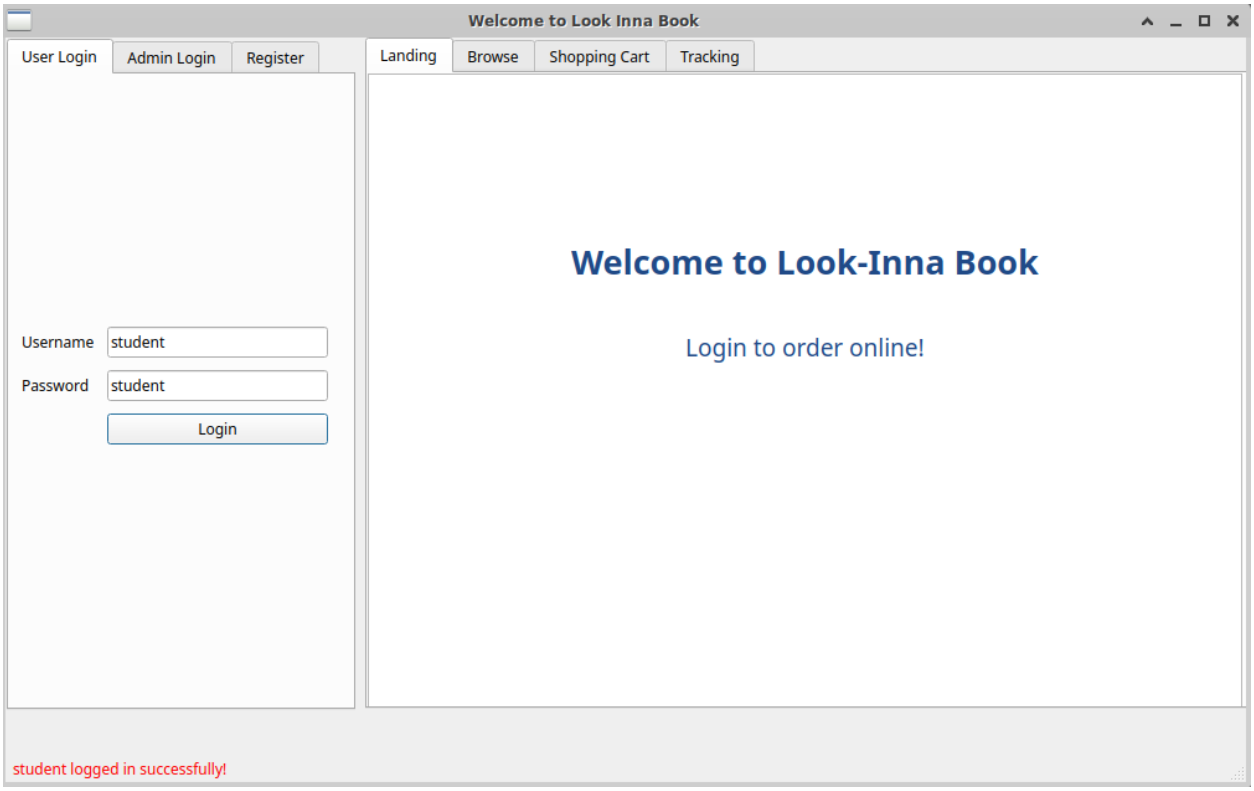
- Login



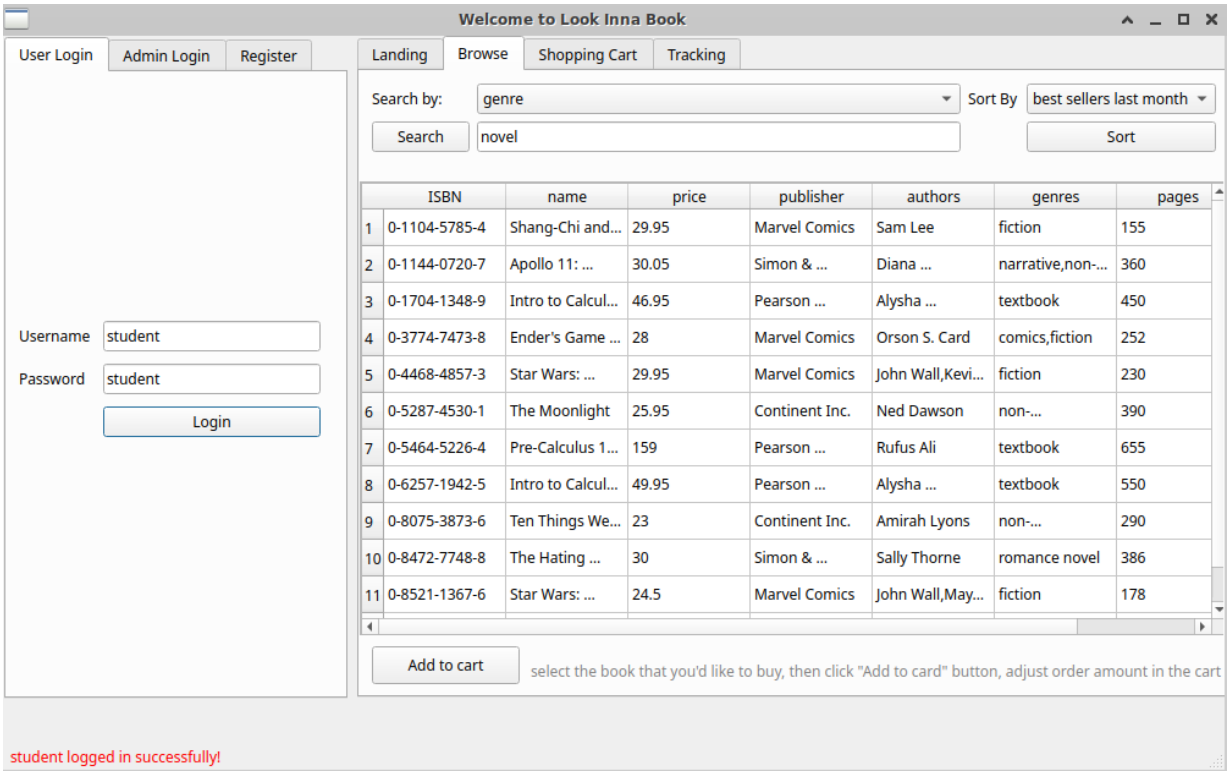
- Register



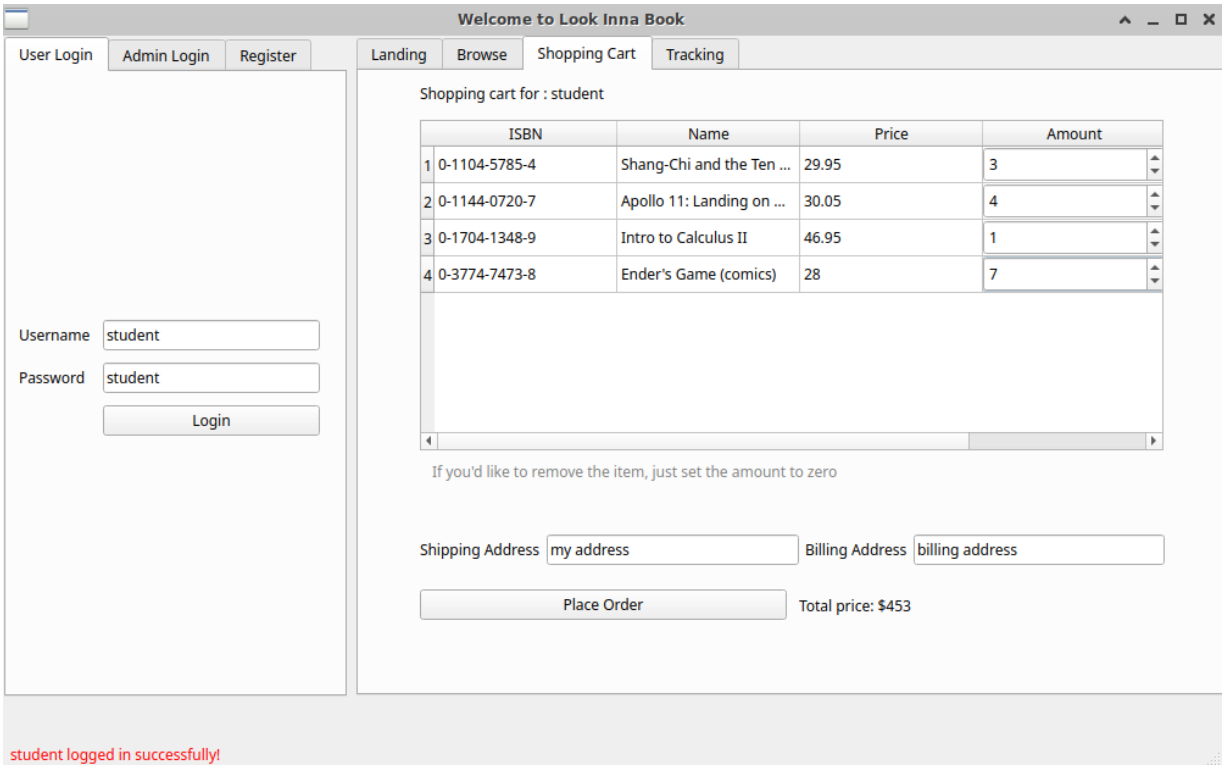
- Landing page



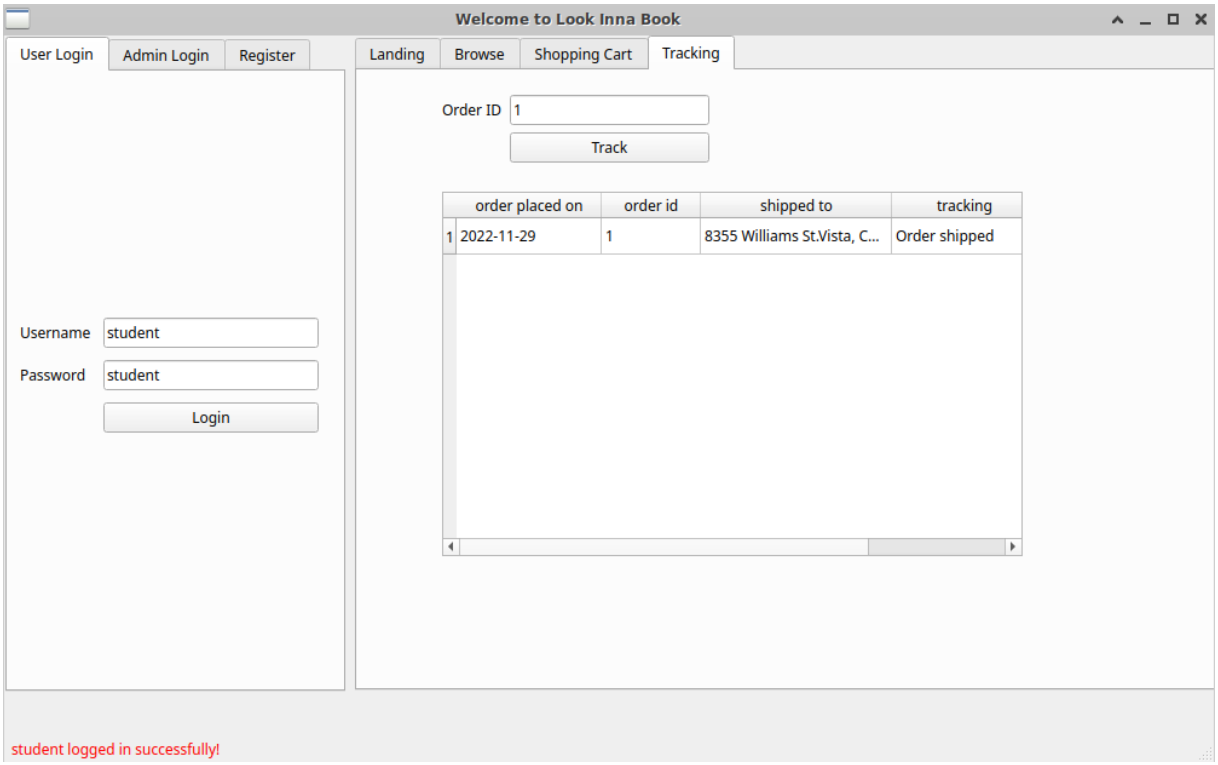
- Browse and search



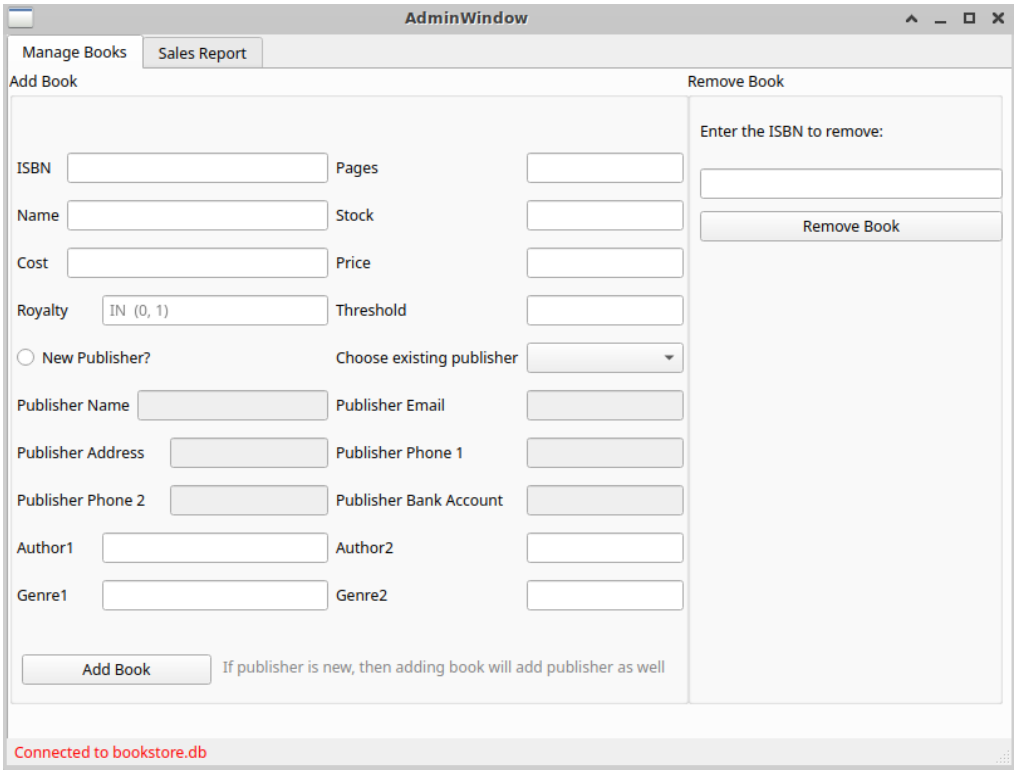
- Shopping cart and place order



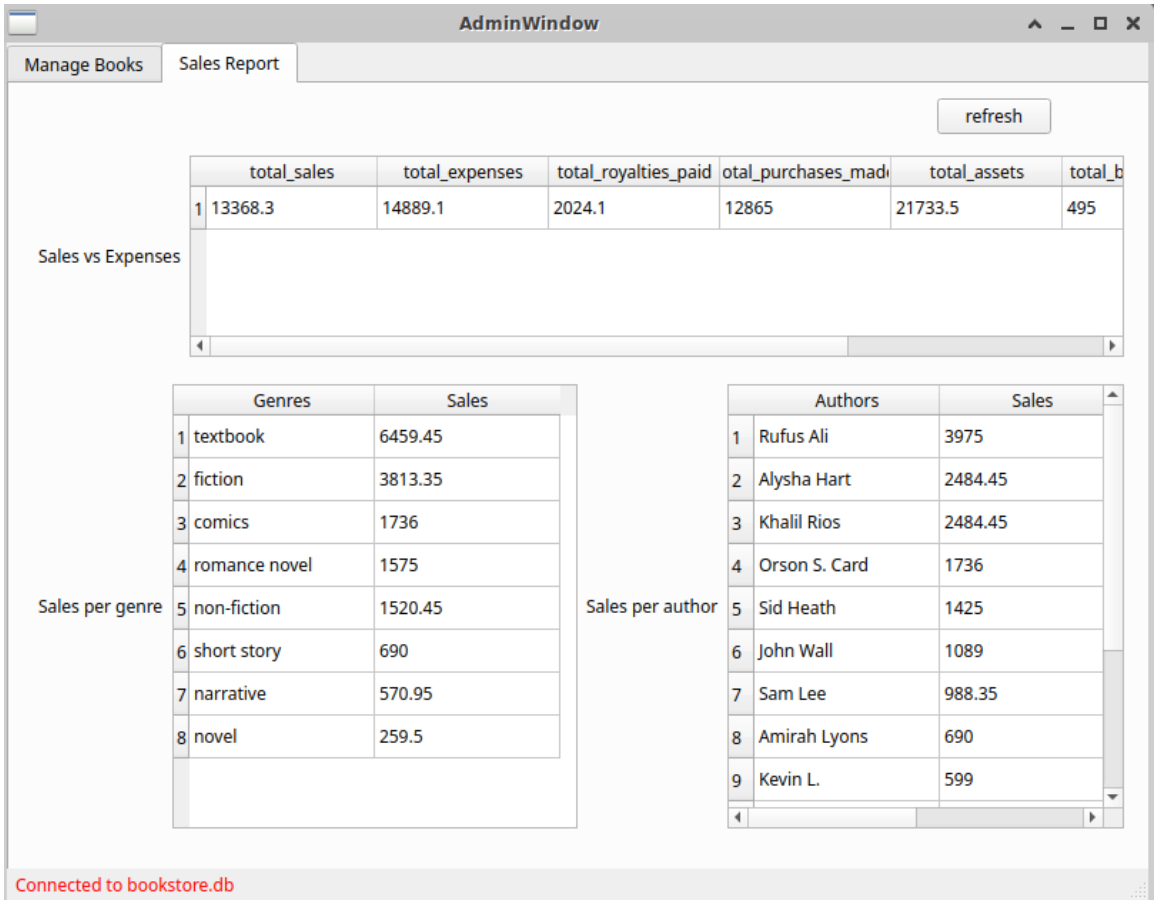
- Tracking an order



- Admin Window – Add/remove books

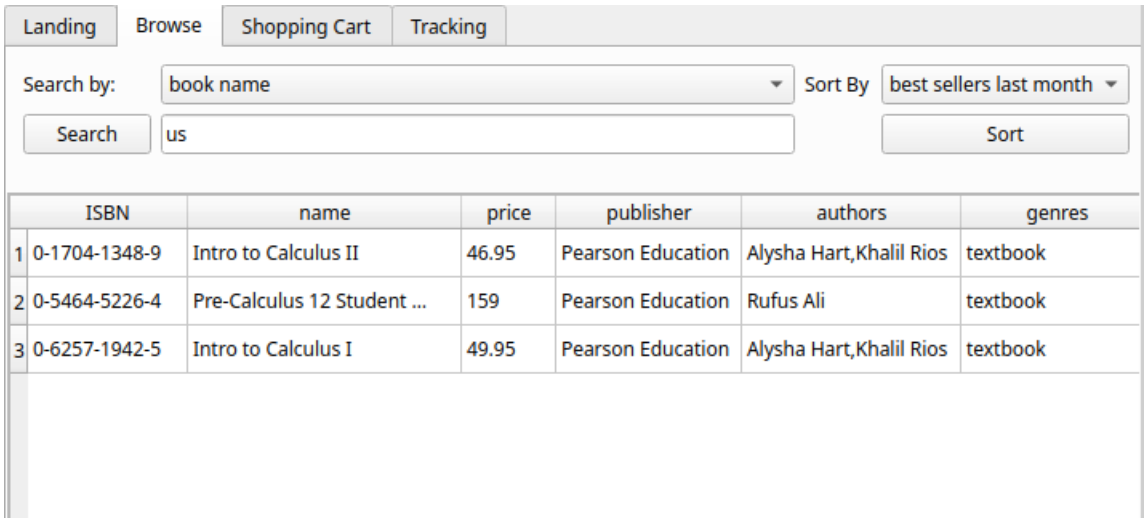


- Sales report



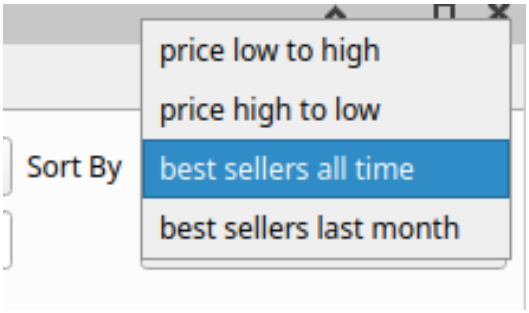
4.4 Bonus Features

- Approximate search



-

- Sort by 4 factors



- Report page with more summarized data

	total_sales	total_expenses	otal_royalties_pai	al_purchases_ma	total_assets	al_book_in_st	total_book_sold
1	13368.3	14889.1	2024.1	12865	21733.5	495	325

5. GitHub Repository

https://github.com/addiexx1/Look-Inna-Book-