Look Inna Book

PROJECT REPORT

By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

Professor: Ahmed El-Roby

COMP 3005
Carleton University

By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

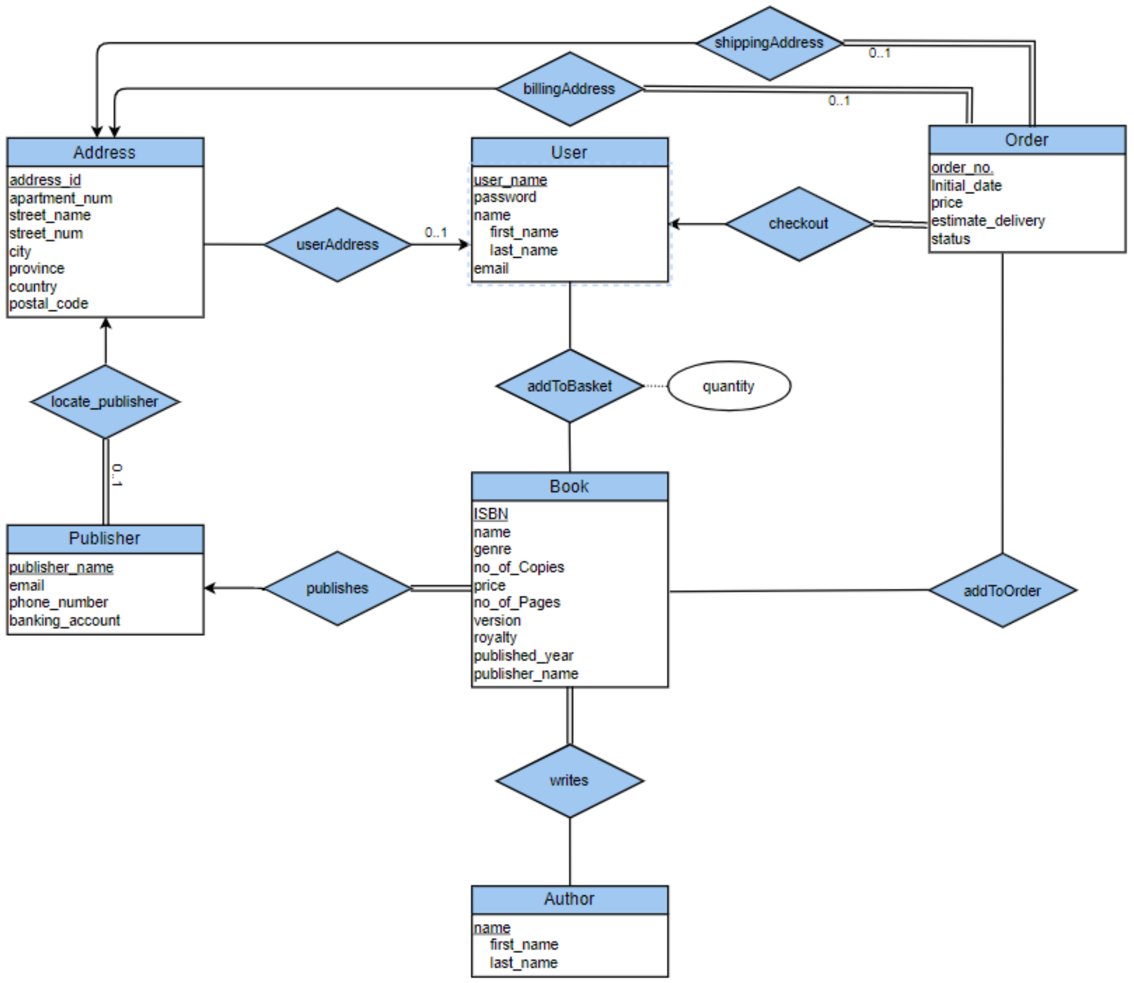
Table Of Contents

By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

1. Conceptual Design (25%)

This section should explain the conceptual design of the database. That is, the ER-diagram of the database for the bookstore and explanation of all the assumptions made in the diagram regarding cardinalities and participation types. Make sure that the assumptions do not contradict with the problem statement in Section 1.

1.1 ER DIAGRAM



By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

1.2 Entities And Attributes

A List of Entities and Their Corresponding Attributes For The Conceptual Design:
Entities: Book, User, Address, Order, Publisher, Author, AddToBasket, Writes, AddToOrder
Attributes:
Book: (ISBN, name, genre, no_of_copies, price, no_of_pages, version, royalty, published_year, publisher_name)
User: (user_name, password, first_name, last_name, email)
Address: (address_id, apartment_num, street_name, street_num, city, province, country, postal_code)
Order: (order_no, initial date, price, estimate_delivery, status, shipping_address_id, billing_address_id, user_name)
Publisher: (publisher_name, email, phone_number, banking account)
Author: (first_name, last_name)
AddToBasket: (user_name, ISBN, quantity)
Writes: (ISBN, first_name, last_name)
AddToOrder: (ISBN, order_no)

- Book
  → The Book entity is a representation of a Book in our database. It corresponds to the primary key attribute ISBN.
- User
  → The User entity is a representation of a User in our database. It corresponds to the primary key attribute user_name.
- Address
  → The Address entity is a representation of the Address in our database. It corresponds to the primary key attribute address_id
- Order
  → The Order entity is a representation of an Order in our database. It corresponds to the primary key attribute order_no.
- Publisher
  → The Publisher entity is a representation of a Publisher in our database. It corresponds to the primary key attribute publisher_name.
- Author
  → The Author entity is a representation of an Author in our database. It corresponds to the primary key attributes first_name and last_name.
- AddToBasket
  → The AddToBasket entity is a representation of a basket in our database. It corresponds to the primary key attributes user_name and ISBN.
- Writes
  → The Writes entity is a representation of a Writes in our database. It corresponds to the primary key attributes ISBN, first_name and last_name.
- AddToOrder
  → The AddToOrder entity is a representation of AddToOrder entity table in our database. It corresponds to the primary key attributes ISBN and order_no.

1.3 Relations

A List Of Relations And Their Corresponding Attributes:

Relations: userAddress, billingAddress, shippingAddress, checkout, locate_publisher, publishes

- UserAddress
  → This relation relates the Address with the User.
- BillingAddress
  → This relation relates the Address with the Order
- ShippingAddress
  → This relation relates the Address with the Order
- Checkout
  → This relation relates the Order with the User.
- Locate_publisher
  → This relation relates the Publisher with the Address. It indicates the Publisher's Address location. Where the Publisher is Located.
- Publishes
  → This relation relates Book with the Publisher. It indicates the Publisher of the Book. It's name, email, phone number and banking account.

By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

1.4. Assumptions Made (Cardinalities, Participation Types etc.)

The following are all the assumptions and their explanation that we have made for this project for the ER-Diagram (cardinalities, participation types etc.):

When making the conceptual design and ER Diagram of the database we made a lot of assumptions.

- Book to Author
  → We assumed that a Book must have an Author, hence the total participation. This is a valid assumption, you can not write a book without an Author. We assumed every Author did not necessarily write a Book, so it is partially participating. We also assumed that a Book can have many Authors, and an Author can have many Books, hence the relationship between Book and Author is many to many.

- Book to Publisher
  → We also assumed that a Book must have a Publisher, hence the Book's total participation with the Publisher. This is a valid assumption, as a Book can not exist (I.E Published in the storebook) without a publisher. Every Publisher must not have published a Book, and a Book can only have 1 publisher. Hence the One to Many relationship between Publisher and Book.

- User to Book
  → We also assumed that when a User adds a Book to the basket, every User does not necessarily have to add a Book to the basket, and every Book does not necessarily have to have been added in a User's basket. Hence we made them a partial participation with each other. A User can add many Books to a basket, and a Book can be in many User's baskets, thus we made the relationship between them Many to Many.

- User to Order
  → Additionally, when User checks out to Order, we assume every Order must have a User checking out, and that every User must not necessarily have an Order they've checked out to. Hence the Partial to Total participation relationship type. We additionally assumed that Every User can have many Orders, however an Order can have just one User. Hence the One to Many relationship type.

- Address to Publisher
  → For the relationship between Address and Publisher. We assumed that the Publisher must have an address, however an address must not necessarily have a Publisher. Hence the Total participation for Publisher and partial participation for Address. We also assumed that a Publisher can have one Address and an Address can only have one Publisher, hence the One to One relationship.

- Address to User
  → For the relationship between Address and User. We assumed that a User can have 1 Address, however an Address can have many Users. Hence the many to 1 relationship. However, a user can still have 1 more address since it can have a different billing address.

- Address to Order
  → For the relationships between Address and Order. An Order can have a shipping as well as a differing billing in some cases. Two orders might have the same shipping and billing but they do not necessarily have to be the same. Every Order must participate with an Address. There can not be an Order without an Address. We assume the relationship is 1 to many. For every 1 address, there can be many orders.

- Book to Order
  → For the relationships between Book and Order. We assumed that the relationship between Book and Order is many to many. Many books can be in many orders. We also assumed that Book and Order are both partial participations as a Book does not necessarily have to be in an Order. And an Order does not necessarily have to be in a Book

2. REDUCTION TO RELATION SCHEMAS

- author = (first_name, last_name)
- address = (address_id, apartment_num, street_name, street_num, city, province, country, postal_code)
- publisher = (publisher_name, email, phone_number, banking_account, address_id)
- user = (user_name, password, first_name, last_name, email, address_id)
- book = (ISBN, name, genre, no_of_copies, price, no_of_pages, version, royalty, published_year, publisher_name)
- Order = (order_no, initial_date, price, estimate_delivery, status, shipping_address_id, billing_address_id, user_name)
- addToOrder = (ISBN, order_no)
- addToBasket = (user_name, ISBN, quantity)
- writes = (ISBN, first_name, last_name)

Note: for the publisher entity, assume a publisher will have a max of one address representing the publishing organization location

By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

3. NORMALIZATION OF RELATION SCHEMAS

- author = (first_name, last_name)
  F = {first_name, last_name → first_name, last_name}

  Using the principle of closure, we find:
  (first_name, last_name)$^+$ = {first_name, last_name)

  Therefore, (first_name, last_name) is trivial and super key.

  Thus, the relation is in BCNF

- address = (address_id, apartment_num, street_name, city, province, country, postal_code)

  F = {address_id → apartment_num, street_name, street_num, city, province, country, postal_code}

  Using the principle of closure, we find:

  address_id$^+$ = {address_id , apartment_num, street_name, street_num, city, province, country, postal_code}

  Therefore, address_id is a super key.

  Thus, the relation is in BCNF

- publisher = (publisher_<u>name</u>, email, phone_number, banking_account, address_id)

  F = {publisher_name → email, phone_number, banking_account, address_id

  email → publisher_name

  banking account → publisher_name

  phone_number→ publisher_name

  address_id → publisher_name **(this assumption is taken upon the note from the previous page)**

  }

  Using the principle of closure, we find:

  publisher_name$^+$ = { publisher_name , email, phone number, banking account, address_id }

Therefore, publisher_name is a super key, and email, banking account, phone, address_id are candidate keys.

Thus, the relation is in BCNF

- user = (user_name, password, first_name, last_name, email, address_id)

  F = {user_name → password, first_name, last_name, email, address_id

  email → user_name}

  Using the principle of closure, we find:

  user_name^+ = { user_name, password, first_name, last_name, email, address_id }

  Therefore, user_name is a super key.

  Thus, the relation is in BCNF

- book = (ISBN, name, genre, no_of_copies, price, no_of_pages, version, royalty, published_year, publisher_name)

  F = {ISBN → name, genre, no_of_copies, price, no_of_pages, version, royalty, published_year, publisher_name }

  Using the principle of closure, we find:

  ISBN^+ = { ISBN, name, genre, no_of_copies, price, no_of_pages, version, royalty, published_year, publisher_name }

  Therefore, ISBN is a super key.

  Thus, the relation is in BCNF

- order = (order_no, initial_date, price, estimate_delivery, status, shipping_address_id, billing_address_id, user_name)

F = { order_no → initial_date, price, estimate_delivery, status, shipping_address_id, billing_address_id, user_name }

Using the principle of closure, we find:

Order_no+ = { initial_date, price, estimate_delivery, status, shipping_address_id, billing_address_id, user_name }

Therefore, order is a super key.

Thus, the relation is in BCNF

- writes = (ISBN, first_name, last_name)

  F = { first_name, last_name → first_name, last_name

  ISBN → ISBN}

  Using the principle of closure, we find:

  (first_name, last_name)$^+$ = {first_name, last_name}

  (ISBN)$^+$ = {ISBN}

  This shows that both Function dependencies are NOT super keys.

  Thus, the relation is NOT in BCNF

  One way to make the relation in BCNF, is by splitting it. Since there is already a relation named used that contains (first_name, last_name), thus a more reliable solution Is through augmenting (first_name, last_name) to (ISBN) to form one super key

  (first_name, last_name, ISBN)$^+$ = {first_name, last_name, ISBN}

  and thus, the relation is now in BCNF

- addToOrder = (ISBN, order_no)

F = {ISBN, order_no → ISBN, order_no}

Using the principle of closure, we find:

(ISBN, order_no)+ = {ISBN, order_no}

Therefore, (ISBN, order_no) is trivial and super key.

Thus, the relation is in BCNF

- addToBasket = (user_name, ISBN, quantity)

  F = {user_name → quantity, ISBN → quantity }

  Using the principle of closure, we find:

  user_name+ = {username, quantity}

  ISBN+ = {quantity}

  This shows that both Function dependencies are NOT super keys.

  Thus, the relation is NOT in BCNF

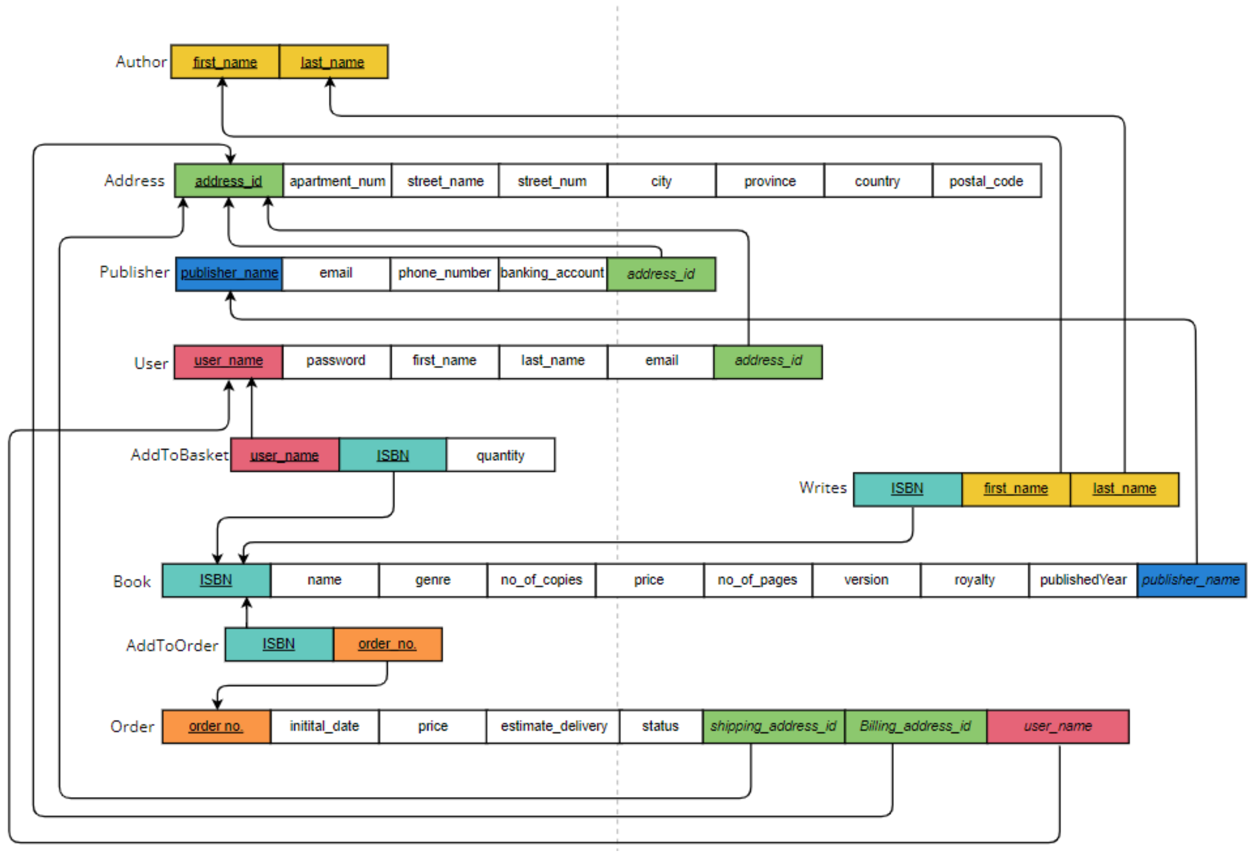
  To make it in BCNF, both user_name and ISBN will be taken together as super key

  (user_name, ISBN)$^{+}$ = {user_name, ISBN, quantity}

  and thus, the relation is now in BCNF

4. DATABASE SCHEMA DIAGRAM (10%)

The resulting Database Schema Diagram after Normalization:



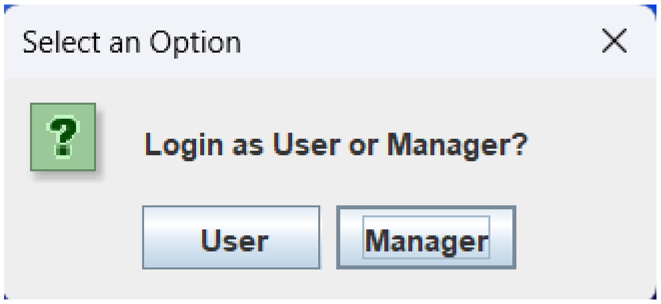
5. Implementation (30%)

Feel free to use whichever programming language(s) for your application. Your application can be a web-based application, a desktop application, or even a command-line application. In this section, you should describe the architecture of your application. That is, what the modules in your application are and how they interact. You are encouraged to include a diagram of the application's architecture and explain (in text) the scenarios of using the application and the workflow of your application. It is mandatory that you use a relational database to store all your data (i.e., no key-value stores). Your application should have two different user interfaces: The first is for the users of the application through which the user can browse and buy books. The second is for the bookstore owners/managers through which they can add/remove books, display reports, etc. (more details about the application's features can be found in Section 1). Include screenshots of your application's two interfaces in different scenarios (e.g., checking out, displaying a report, etc.)

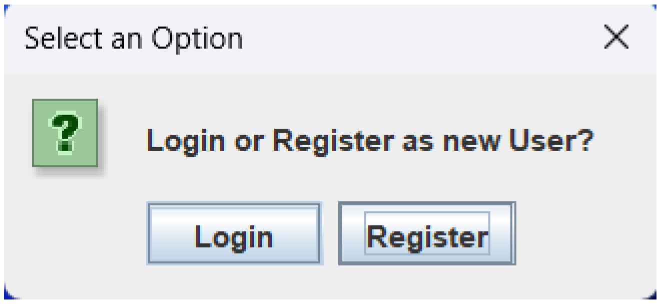
Our Bookstore implementation aims to provide a user-friendly graphical interface for managing our bookstore database.The GUI implementation allows Managers to easily view, add, update, and delete books, users, authors, publishers, sales and reports in the database. It allows the users to easily search for specific books using various criteria and to easily add and remove books from their basket and order. The intuitive and responsive design of the interface makes it easy for managers to quickly perform the most common tasks without having to write complex SQL queries.

5.1 Login Screen

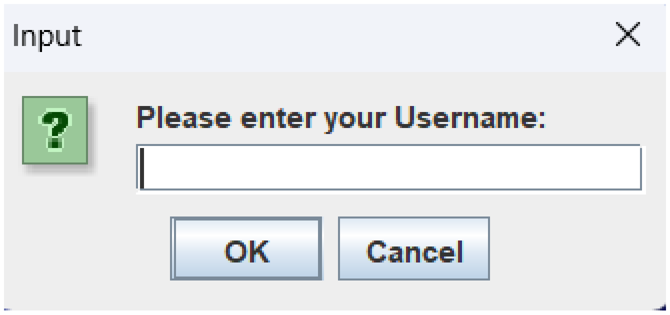
When opening the jar file, the first interface that comes up is the following below. Asking the user to login as a User or Manager. Either Login as a User, or a Manager:

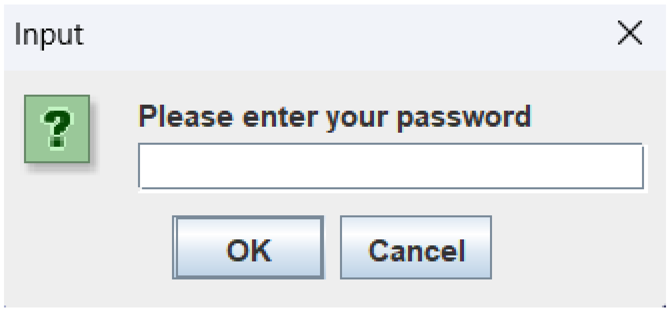


If User option is pressed, either Login or Register:



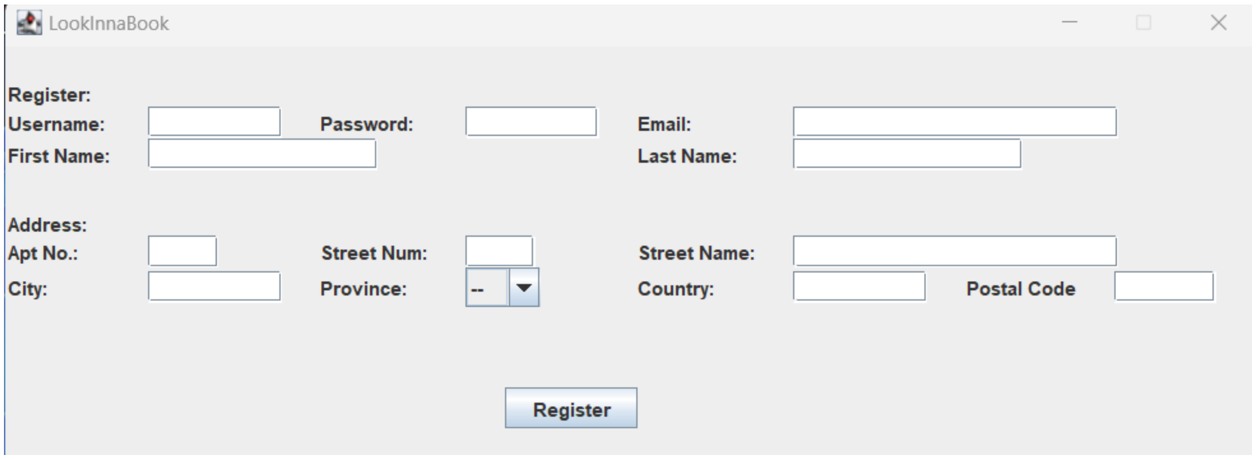
You can then login by entering your Username and Password:
(Note: We have a default user named “admin” with password “admin”).





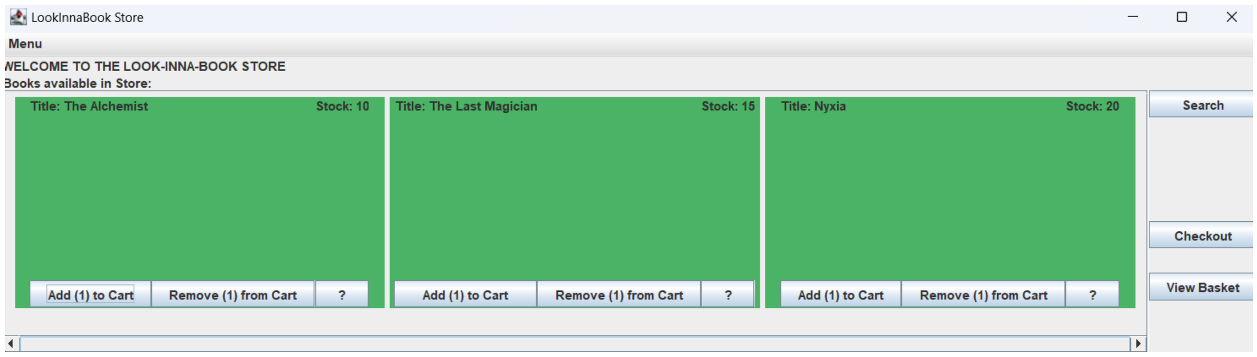
5.2 Registration Screen

If Register pressed, filled in the specified input fields to create a new user account:
(Note: User can not enter an already taken Username that is in the Database).



By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

5.3 The User GUI

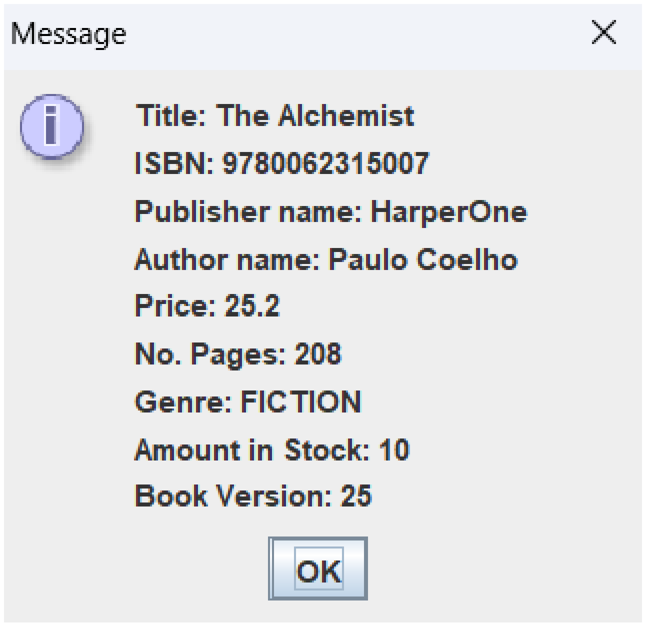

LookInnaBook Store

Menu

WELCOME TO THE LOOK-INNA-BOOK STORE

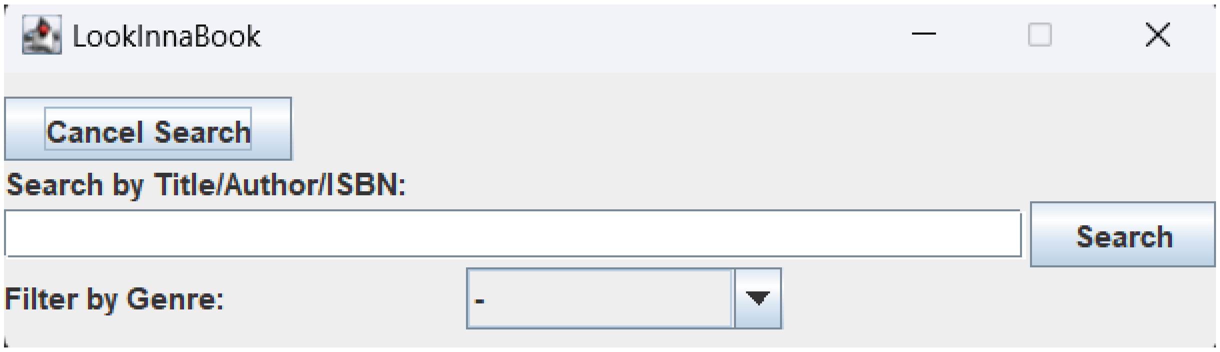
Books available in Store:

If the question mark on a book is pressed, it's information is shown as below:


Message

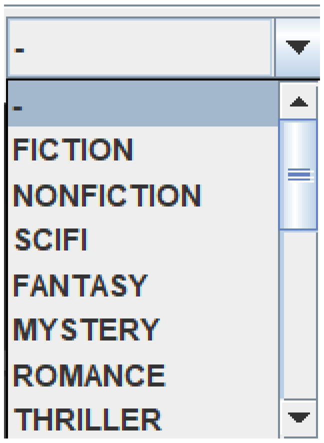
Title: The Alchemist
ISBN: 9780062315007
Publisher name: HarperOne
Author name: Paulo Coelho
Price: 25.2
No. Pages: 208
Genre: FICTION
Amount in Stock: 10
Book Version: 25

If User presses search the following interface is shown. With a filter by genre combo box as well:


LookInnaBook

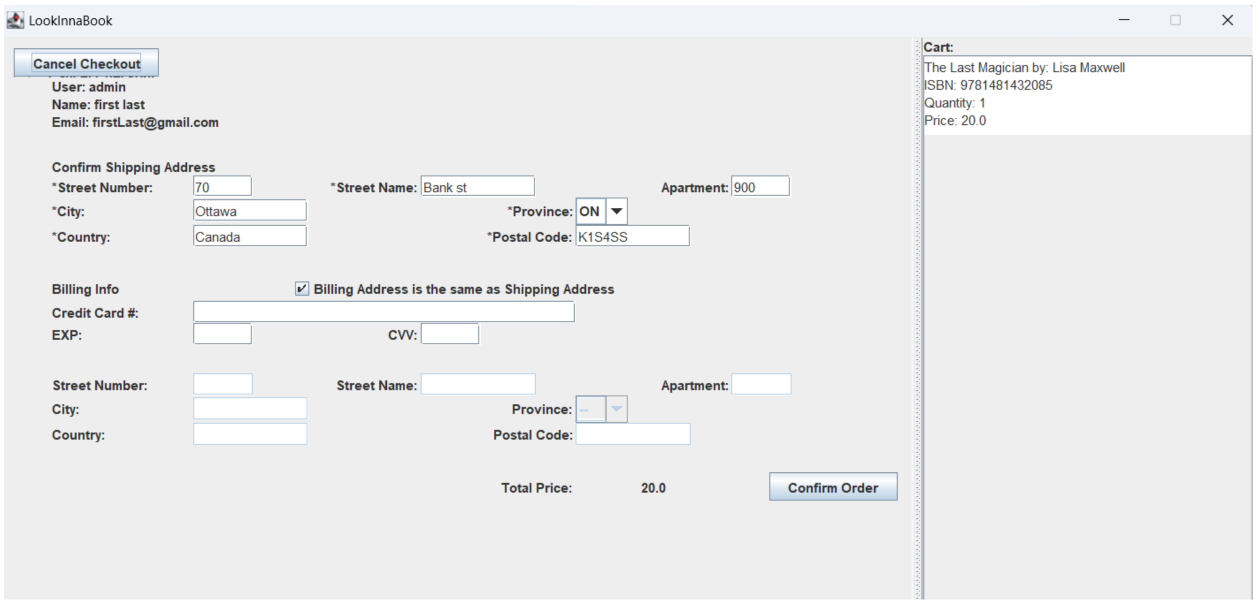
Search by Title/Author/ISBN:

Filter by Genre: -



5.4 Checkout Screen

The following below is the User's Checkout Screen. Input all required information and then order, checkout and ship a book to the user's given address.



User: admin
Name: first last
Email: firstLast@gmail.com

Confirm Shipping Address

*Street Number:	70	*Street Name:	Bank st	Apartment:	900
*City:	Ottawa	*Province:	ON		
*Country:	Canada	*Postal Code:	K1S4SS		

Billing Info [✓] Billing Address is the same as Shipping Address

Credit Card #:			
EXP:		CVV:	

Street Number:		Street Name:		Apartment:	
City:		Province:	--		
Country:		Postal Code:			

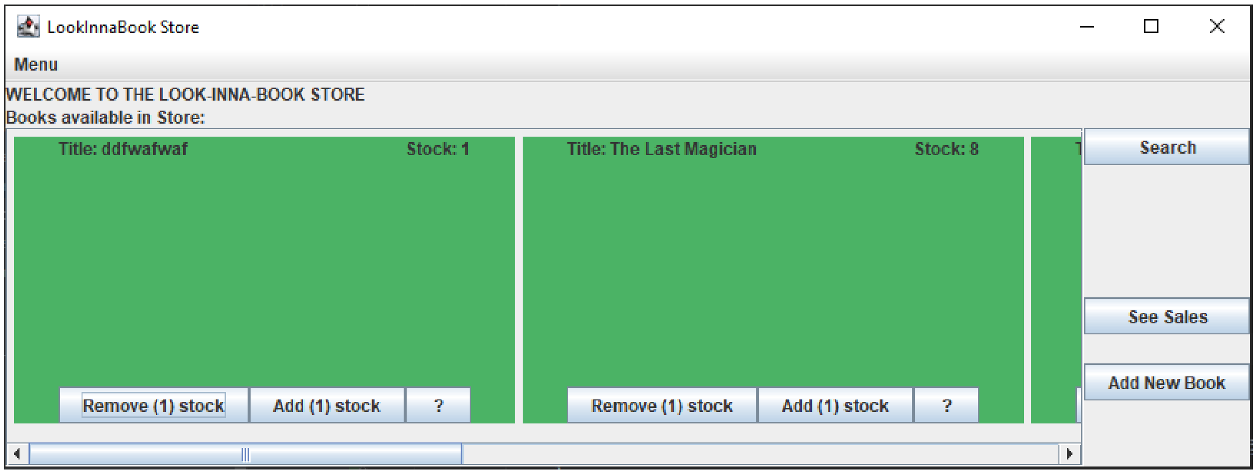
Total Price: 20.0

Cart:
The Last Magician by: Lisa Maxwell
ISBN: 9781481432085
Quantity: 1
Price: 20.0

By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

5.5 Manager GUI

The following below is the Manager's GUI Screen. The manager can add new books into the bookstore and see sales.

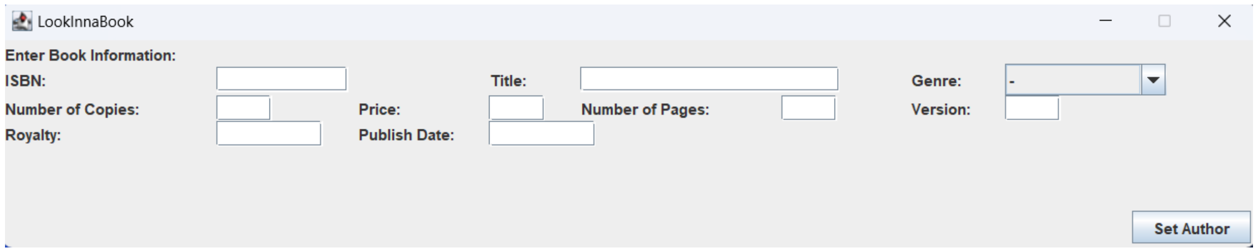


WELCOME TO THE LOOK-INNA-BOOK STORE
Books available in Store:

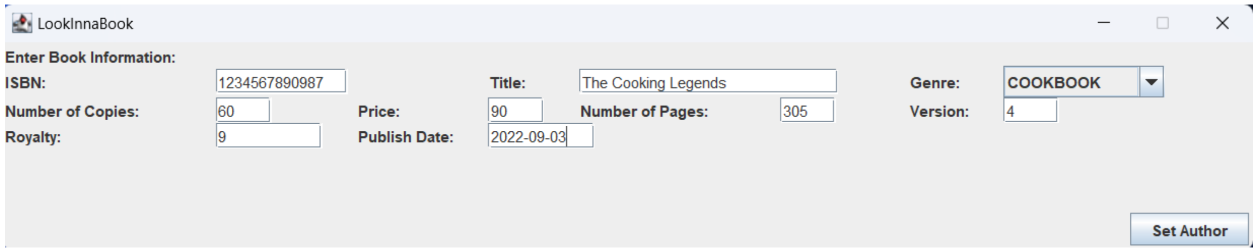
Title: ddfwafwaf Stock: 1

Title: The Last Magician Stock: 8

From the menu above, the manager can then Add New Books:



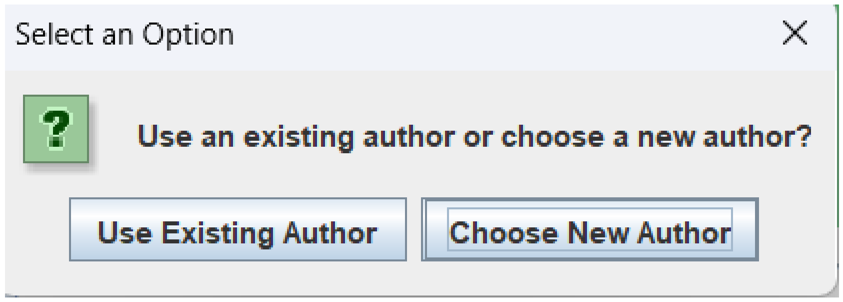
With the proper values input as shown below:



Enter Book Information:

ISBN:	1234567890987	Title:	The Cooking Legends	Genre:	COOKBOOK
Number of Copies:	60	Price:	90	Number of Pages:	305
Version:	4	Royalty:	9	Publish Date:	2022-09-03

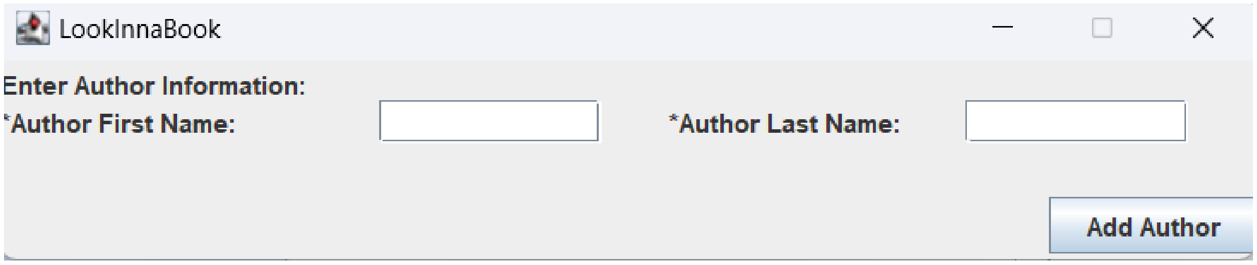
When Set Author is pressed. Choose an exist Author, or a new one:



Use an existing author or choose a new author?

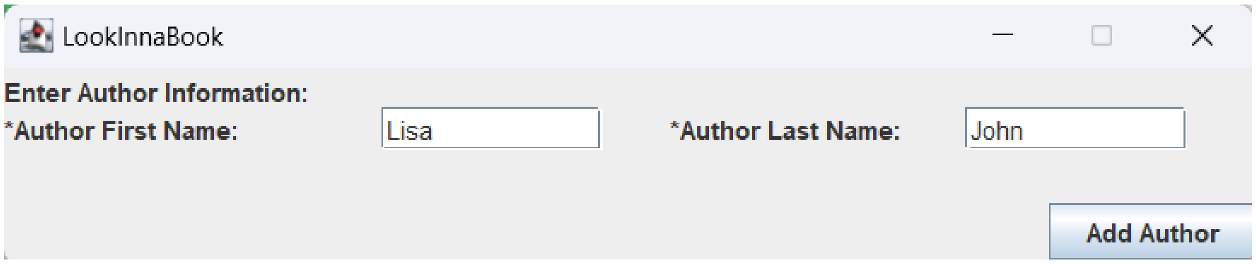
By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

If choosing a new author, Add an Author with their corresponding first and last name


LookInnaBook

Enter Author Information:
*Author First Name:
*Author Last Name:

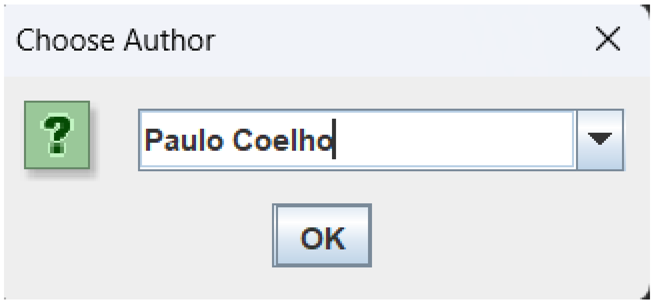
Add Author


LookInnaBook

Enter Author Information:
*Author First Name: Lisa
*Author Last Name: John

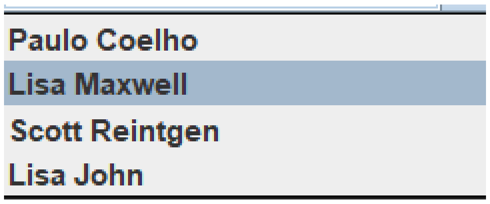
Add Author

If choosing an existing author, choose as below:

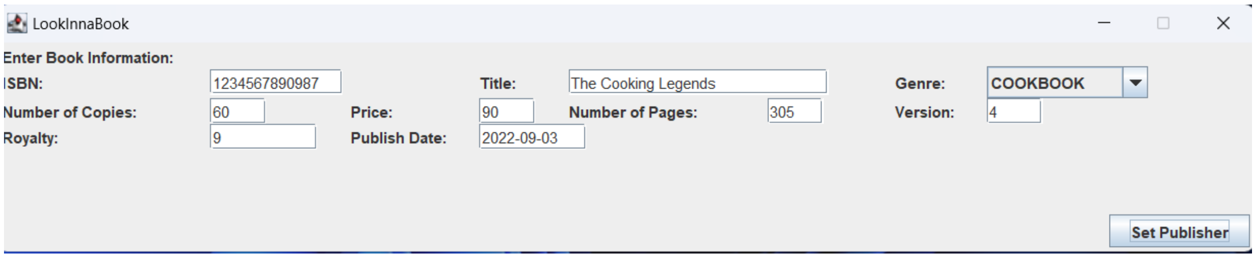

Choose Author

Paulo Coelho

OK


Paulo Coelho
Lisa Maxwell
Scott Reintgen
Lisa John

The author will be added, and you can then set a publisher on the bottom left:

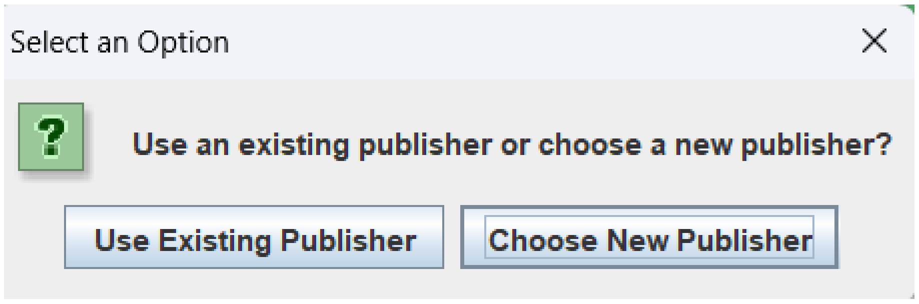

LookInnaBook

Enter Book Information:
ISBN: 1234567890987
Title: The Cooking Legends
Genre: COOKBOOK
Number of Copies: 60
Price: 90
Number of Pages: 305
Version: 4
Royalty: 9
Publish Date: 2022-09-03

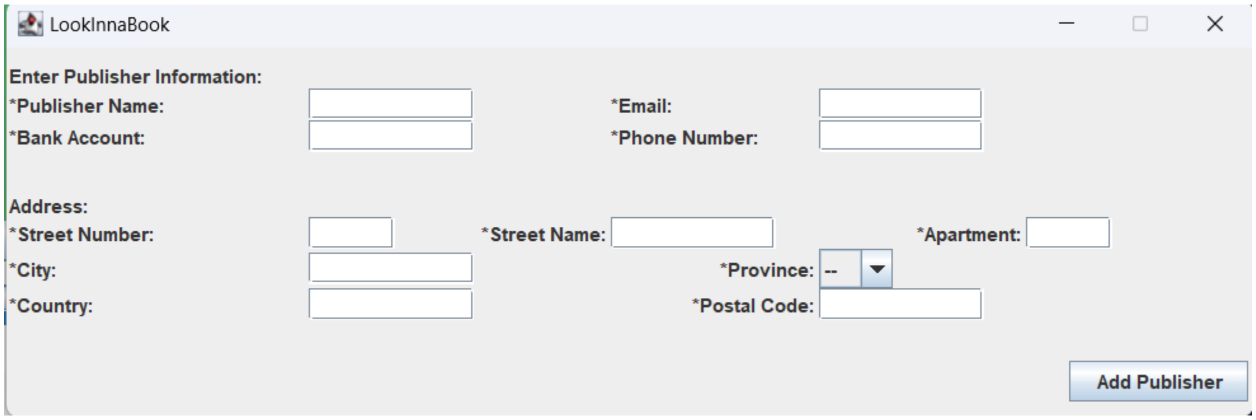
Set Publisher

By: Amr Abdelazeem, Shahrik Amin, Patrick Ma

Use and existing publisher, or choose a new one:



If "Choose New Publisher" is pressed, the following popup is shown:



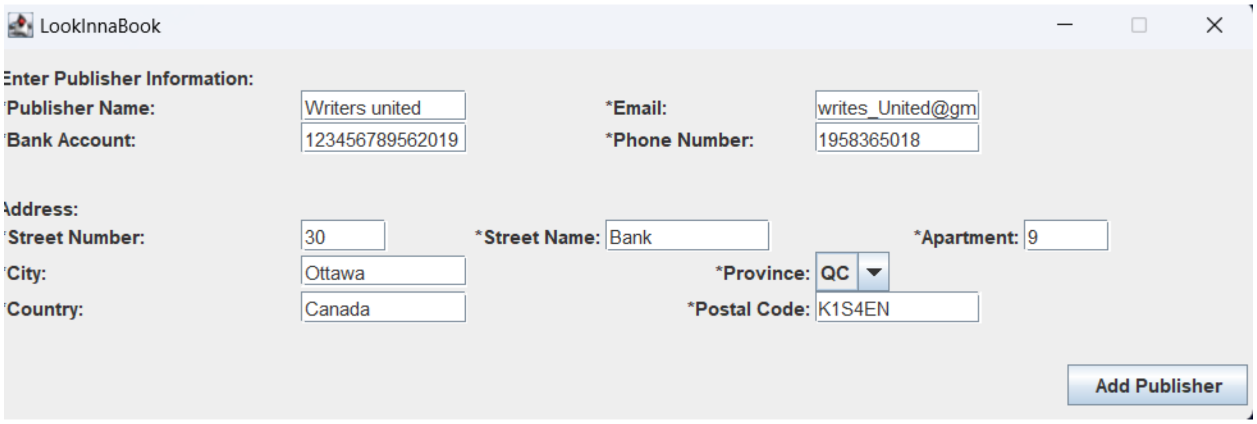
Enter Publisher Information:

*Publisher Name:		*Email:			
*Bank Account:		*Phone Number:			

Address:

*Street Number:		*Street Name:		*Apartment:	
*City:		*Province:	--		
*Country:		*Postal Code:			

With the proper input values:



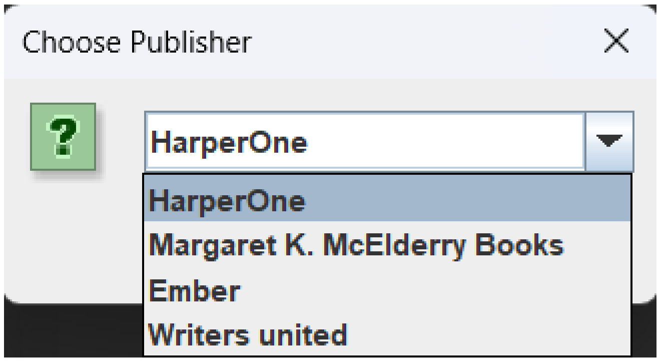
Enter Publisher Information:

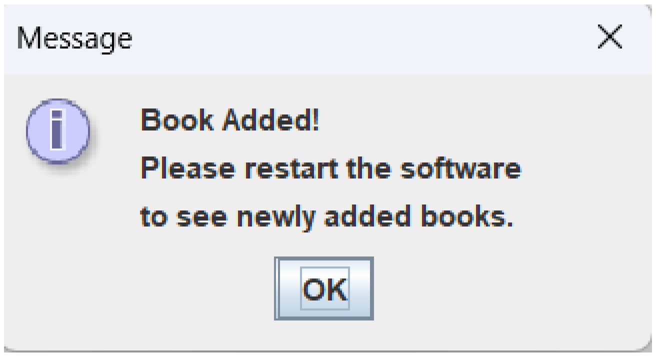
*Publisher Name:	Writers united	*Email:	writes_United@gm		
*Bank Account:	123456789562019	*Phone Number:	1958365018		

Address:

*Street Number:	30	*Street Name:	Bank	*Apartment:	9
*City:	Ottawa	*Province:	QC		
*Country:	Canada	*Postal Code:	K1S4EN		

If instead "Choose Publisher" is pressed, choose the proper publisher as below:




Message

Book Added!
Please restart the software
to see newly added books.

5.6 Scenarios

1. User tries to login while not registered or with a wrong password. In this scenario, a popup interface with “incorrect username or password” comes up.

2. User tries to register with an already registered user_name or email. In this case, the system prevents the user from registering with this user_name or email

3. A User tries to remove a Book that does not exist in their basket. In this case, the system will not allow the user to remove this book.

4. The User tries to search up for a Book that does not exist in the store. In this scenario, the search result will be 0.

5. The User closes the search screen. In this case the User will be taken back to the main screen of the programs.

6. The User closes the checkout screen. In this case the User will be taken back to the main screen of the programs.

7. The Manager tries to add a Book that already exists. In this case the book quantity will be updated corresponding to the amount the manager needs to add.

8. The Manager tries to add an Author that already exists. In this case the Manager will receive a message saying the Author already exists.

9. The Manager tries to add a Publisher that already exists. In this case the Manager will receive a message saying the Author already exists.

6 Bonus Features (Optional Section - Up to 15%)

6.1 Genre Search

Implementing a genre search feature is a valuable bonus addition to our book management system. This feature allows users to easily search for books based on their genre, making it easier for users to find books that match their interests and preferences. For example, if the User picks the “Fantasy” genre. The search will show all books under the “Fantasy” genre in the bookstore database. By providing this capability, we have demonstrated a commitment to making our system as user-friendly and useful as possible. In addition, genre search can also be a useful tool for bookstores and libraries to help organize and categorize their collections. We have gone beyond what was asked of us and I believe that implementing this feature warrants consideration for a bonus mark.

6.2 Graphical User Interface

In our project description, there was no mention of a Graphical User Interface (GUI) and how it should be implemented. However, we decided to implement one with many features that enhance the user experience. For example, we added a scroll bar so the user can easily browse through the books, a filter by genre combo box to allow the user to search by genre, and we even implemented a feature that allows the user to close the app using the X button. And much more implementation in relation to the GUI. We believe that these additional features add value to our project and should be considered for bonus marks.

6.3 Approximate Search

There is no need or requirement of an approximate search as per the project description. However, we have implemented an approximate search feature such that when the User types in the search text it shows all books in the bookstore database with a similar title name. Implementing an approximate search feature is a valuable addition to our book management system. This feature allows users to search for books based on approximate spellings or keywords, making it easier for Users to find books even if they are not sure of the exact title or author. This is especially useful in a book store setting, where users may not always have all the information about a book at hand. By providing this capability, we have shown a commitment to making our system as user-friendly and accessible as possible. We believe that implementing this feature warrants bonus marks.

6.4 Adding Publisher

In our project description, there was no requirement to implement a feature allowing the manager to add a publisher without adding books. However, we decided to include this feature in our project. We believe that this additional feature adds value to our project and demonstrates our dedication to going above and beyond the requirements. Furthermore, the ability to add publishers without adding books allows for greater flexibility and efficiency in managing the system. We believe that implementing this feature warrants our group for bonus marks.

6.5 Adding an Author

In our project description, there was no requirement to implement a feature allowing the manager to add an author without adding books. However, we decided to include this feature in our project. We believe that this additional feature adds value to our project and demonstrates our dedication to going above and beyond the requirements. Furthermore, the ability to add publishers without adding books allows for greater flexibility and efficiency in managing the system. We believe that implementing this feature warrants our group for bonus marks.

6.6 Update A Specified Number Of Book Quantity

We have implemented a feature that was not originally specified in the project description. The ability to update the quantity of a specified book is a valuable feature that can help managers efficiently and accurately update the number of books the store has in stock. This is especially useful in a book store setting, where it is important to know how many copies of a particular book are available for purchase. By implementing this feature, we have demonstrated a keen understanding of the needs of your users and have gone above and beyond the requirements of the project. As a result, we believe we should be considered for bonus marks for our efforts.

6.7 Update Shared Royalty Of The Publisher

Implementing the ability to update the shared royalty of a publisher is a useful feature that can help the Manager manage the financial aspects of their bookstore. This feature allows the Manager to track and adjust the percentage points of royalties that are shared with a particular publisher. This ensures that royalty agreements are accurately reflected in the system. By implementing this feature, we have shown a solid understanding of the needs of Book publishers and have demonstrated a commitment to providing a comprehensive and user friendly system and going above what has been asked of us. As a result, I believe we should be considered for bonus marks for our extra efforts.

6.8 Update Publisher Address

Implementing the ability to update the address of a publisher is a valuable addition to our book management system. This feature allows users to easily keep track of the contact information for the publishers they work with, ensuring that their records are accurate and up-to-date. By providing this capability, we have demonstrated a commitment in providing a comprehensive and user-friendly database system for managing book publishing information. In addition, being able to update publisher addresses is useful for a bookstore, as it allows them to accurately track the locations of the publishers they work with. Overall, we believe that implementing this feature warrants consideration for bonus marks.

7 How To Run

1. You can clone the project onto your desired IDE from the GitHub repository
2. Open a terminal window or command prompt and navigate to the directory where you want to clone the project.
3. Use the ‘git clone’ command to clone the project from GitHub.
   git clone https://github.com/PatrickMa864/Comp3005Project
4. Open postgresSQL and created a new dataBase with the name “COMP3005_Project”
5. Right click on the dataBase created and choose query Tools from the shown menu.
6. Copy paste the DDL file on github to the Query terminal shown and hit run.
7. Open a new Query window or delete all the content on the current one.
8. Now copy paste the DML file and hit run again.
9. You should have your dataBase working now
10. Once the database has been created and you have the project cloned, connect to the database by going to ‘DatabaseQueries.java’ and modifying the connection parameters in the code to match your own PostgreSQL server settings. You will need to provide the hostname and port of the server, as well as a valid username and password with access to the database. For example, if your PostgreSQL server is running on localhost on port 5432 with a username of ‘postgres’ and a password of ‘password’, change it to these values accordingly.
11. Run the main method in the 'LookInnaBookFrame.java' class to run the project.

8 Github Repository

Link of Github Repository: https://github.com/PatrickMa864/Comp3005Project