Order Processing System
BOOKSTORE

DataBase project

—

Names	ID
Afnan Mousa	15
Enas Morsy	20
Sara Mohammad	31
Shimaa Kamal	34

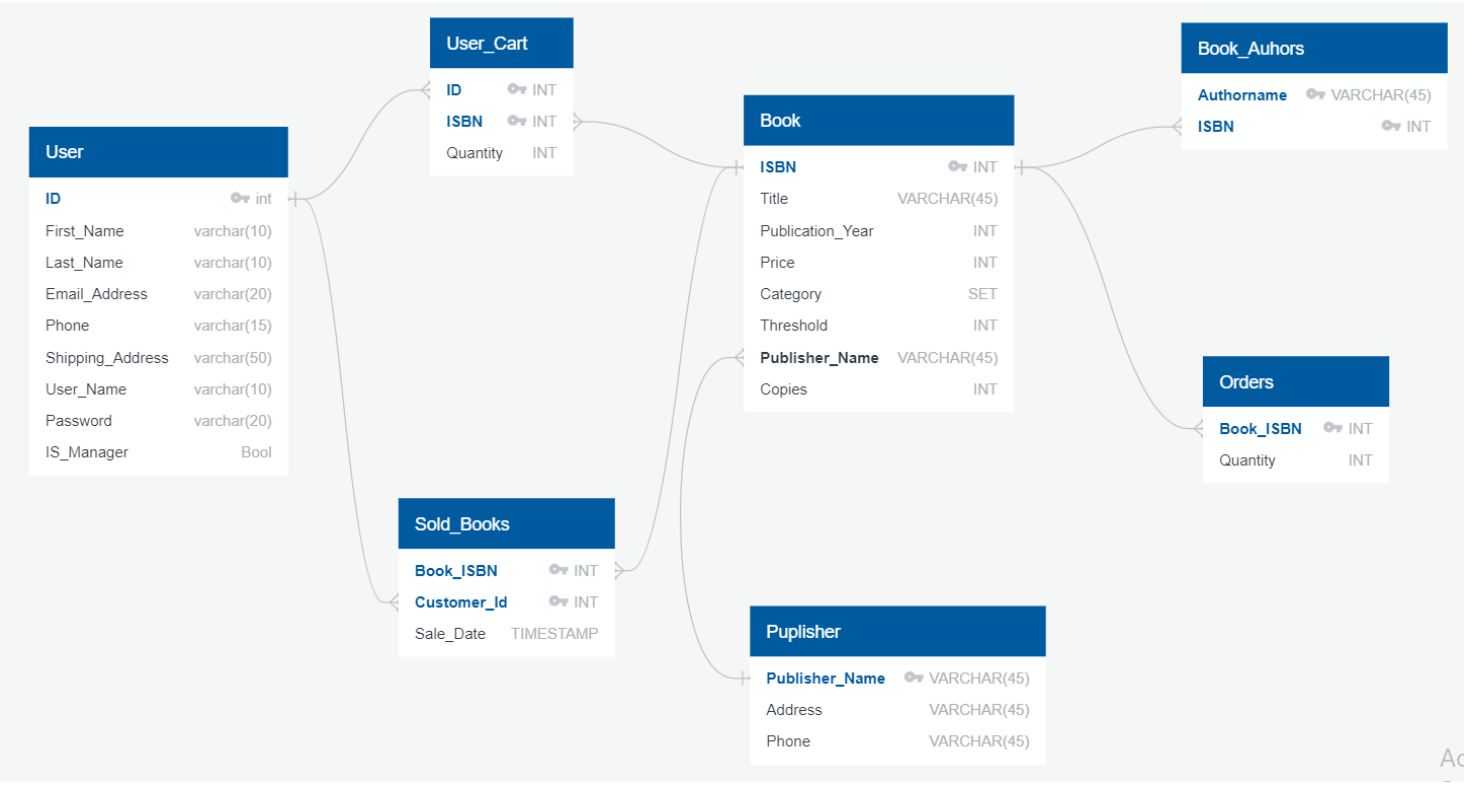
★ TO RUN THE CONNECTION :

Change the username and password in **Connection.java** class in lines 11 , 12.

Database:

1. ERD diagram

➢ Simple MySQL generation needs to be adjusted in order to become user readable.



2. Procedures

Manager's access:

- **Add_Book**
  - The main purpose for this procedure is to add a new book, in which the user enters the properties of the new book along with a threshold.
  - Do that by making procedure Add_Book take all attributes of BOOK Table which are not null these attribute are :
    - ISBN : The Book's ID number.
    - Title : The title of the Book.
    - Publication_Year : the year which book was published .
    - Price : The selling price.
    - Category : is a set of ('Science','Art','Religion','History','Geography').
    - Threshold : The minimum quantity in stock to be maintained for that book.
    - Publisher Name.
    - Copies : Number of Copies which the user needs to add or remove from a specific Book.
  - Use SQL command insert into BOOK values(ISBN,Title, Publication_Year, Price, Category, Threshold, PUBLISHER_Name, Copies) (-- add all attributes which parse as parameter to the procedure --).
  - After that the new Book was added to the database of the BOOK Table.

- **Modify_Book**
  - The main purpose for this procedure is to update an existing book,so that the user can first search for the book then he does the required update.
  - Do that by making procedure Modify_Book take all attributes of BOOK Table which are not null these attribute are :
    - ISBN : The Book's ID number.
    - Title : The title of the Book.
    - Publication_Year : the year which book was published.
    - Price : The selling price.
    - Category : is a set of ('Science','Art','Religion','History','Geography').
    - Threshold : The minimum quantity in stock to be maintained for that book.
    - Publisher Name.
    - Copies : Number of Copies which the user needs to add or remove from a specific Book.
  - Use SQL command Update BOOK as B set:
    - At this point some attribute which parse can be null when the user did not need to modify it and other attributes have the value which the user inserts it to update the values.
    - Use the COALESCE function which takes a list of values and returns the first non null element. In our case take two values first is the attribute which user parses to the procedure another the attribute of BOOK Tuple when the where condition in the update command achieved.
    - Finally the procedure updates the BOOK Tuple by values not null where BOOK.ISBN=ISBN which parse to the procedure.
  - The main assumption is that IF (Copies is null) THEN set Copies = 0.
    - Because the new number of copies always equal BOOK.Copies +Copies which Copies can be negative in case the book sold and can be positive if book bought.

- **ADD_ORDER**
  - The main purpose for this procedure is to add a new Order, in which the user enters the properties of the new order.
  - Do that by making procedure ADD_ORDER take all attributes of ORDERS Table which are not null these attribute are :
    - BOOK_ISBN: The Book's ID number.
    - Quantity : constant quantity represents the number of copies which need to be added to a specific Book.
  - Use SQL command Insert into ORDERS values (BOOK_ISBN, Quantity).
  - After that the new Order was added to the database of the ORDER Table.

- **CONFIRM_ORDER**
  - The main purpose for this procedure is that the quantity of the book is stored automatically in the BOOK Table and increases the BOOK.Copies by the quantity specified in the order.
  - That occurs when the user can confirm an order when receiving the ordered quantity from the book's publisher.
  - Do that by making procedure CONFIRM_ORDER take only one attribute:
    - ISBN : The Book's ID number which is used to get the number of Quantity from ORDERS Table.
  - Firstly declare the variable CNT to indicate the number of Quantity and use SQL command.
  - set CNT := (select Quantity from ORDERS as O where O.BOOK_ISBN = BOOK_ISBN);
  - After searching by BOOK_ISBN and getting the number of Quantity and although set it in the CNT variable .
  - CALL Modify_Book(BOOK_ISBN,null,null,null,null,null,null,CNT), to update the number of Copies in specific BOOK Tuple determined by BOOK_ISBN.

- **DELETE_ORDER**
  - The main purpose for this procedure is to delete a new Order.
  - Do that by making procedure DELET_ORDER take only one attribute from ORDERS Table :
    - ISBN : The Book's ID number which is used to delete the specific tuple from ORDERS Table.
  - Use SQL command delete from ORDERS as O where O.BOOK_ISBN =BOOK_ISBN.
  - After that the specific Order was removed from the database of the ORDER Table.

- **SEARCH_ON_BOOK**
  - The main purpose for this procedure is to allow the user to search for a book by ISBN, and title. The user can search for books of a specific Category, author or publisher.
  - Do that by making procedure SEARCH_ON_BOOK take attributes of BOOK Table and BOOK_AUTHORS Table these attribute are :
    - ISBN : The Book's ID number.
    - Title : The title of the Book.
    - Category : is a set of ('Science','Art','Religion','History','Geography').
    - Publisher Name.
    - Author Name.
  - First check IF ISBN = 0 THEN SET ISBN = NULL .
  - After that use SQL command to select multiple attributes from two table BOOK Table and BOOK_AUTHORS Table using Natural join,
    select
    B.ISBN,B.Title,B.Publication_Year,B.Price,B.Category,B.PUBLISHER_Name,BA.Authorname
    from BOOK as B natural join BOOK_AUTHORS as BA.
  - In the where condition check if the value of attribute is null **XOR** using the function build in FIND_IN_SET which take (string , string_list) and check :
    - If *string* is not found in *string_list*, this function returns zero.
    - If *string* or *string_list* is NULL, this function returns NULL
  - Make this function with all attributes in the where condition and use OR between the conditions but use only one AND between ISBN and Title as required in the PDF .
  - And finally print all attributes located in the select command if the where condition occurs for each Tuple.

Shopping Cart operations:

- **ADD_ITEM**
  - Used to insert a new book into user's shopping cart through passing essential parameters which are:
    - USER_ID: the id of the current user who performs the action
    - BOOK_ID: the ISBN of the book the user wants to get
    - QUANTITY: number of book's copies user wants to get
  - If the passed parameters are valid then new tuple is added to USER_CART table that contains the contents of all users' shopping carts
  - This is done through using the query:
    - INSERT INTO USER_CART
    - VALUES (USER_ID , BOOK_ID , QUANTITY);

- **REMOVE_ITEM**
  - Used to remove a book from user's shopping cart through passing essential parameters which are:
    - USER_ID: the id of the current user who performs the action
    - BOOK_ID: the ISBN of the book the user wants to remove
  - If the passed parameters are valid then the corresponding tuple is removed from USER_CART table
  - This is done through using the query:
    - delete from USER_CART as C
    - where C.ISBN = BOOK_ID AND C.ID = USER_ID;

- **REMOVE_ALL_ITEMS**
  - Used to empty the user's shopping cart completely through passing:
    - USER_ID: the id of the current user who wants to remove all saved books in his cart
  - If his ID is valid then all tuples belonging to this user are removed from USER_CART table
  - This is done through using the query:
    - delete from USER_CART as C
    - where C.ID = USER_ID;

- **VIEW_ITEMS**
  - Used to enable the user to view all the books in his cart at any time through passing the parameter:
    - USER_ID: the id of the current user who wants to review his shopping cart
  - If his ID is valid then data of all books saved by this user is presented
  - To get such behaviour, USER_CART contains only book_ISBN, user_ID and required QUANTITY, but no far information about this book
  - So Natural_JOIN is used between BOOK table and USER_CART table to get tuples holding all books information that belong to this user shopping_cart
  - This is done through using the query:
    - SELECT *
    - FROM BOOK_NATURAL JOIN USER_CART
    - WHERE USER_CART.ID = USER_ID;

- **GET_PRICES**
  - Used to show the price of each individual book in the user's shopping cart through passing the parameter:
    - USER_ID: the id of the current user who performs the action
  - If his ID is valid then the title and the price of each book are printed out to the user
  - To get such behaviour, USER_CART contains only book_ISBN, user_ID and required QUANTITY
  - So Natural_JOIN is used between BOOK table and USER_CART table to be able to access each book title and original price
  - The individual price depends on the book original price and the required quantity
  - This is done through using the query:
    - SELECT Title, Price*QUANTITY
    - FROM BOOK_NATURAL JOIN USER_CART
    - WHERE USER_CART.ID = USER_ID;

- **TOTAL_PRICE**
  - Used to show the total price of all books in the current user's shopping cart through passing the parameter:
    - USER_ID: the id of the current user who performs the action
  - If his ID is valid then he gets the total price of his saved books
  - To get such behaviour, USER_CART contains only book_ISBN, user_ID and required QUANTITY
  - So Natural_JOIN is used between BOOK table and USER_CART table to be able to access each book original price
  - Then aggregation function, SUM, is applied to the selected column to get the overall price
  - This is done through using the query:
    - SELECT SUM(Price) AS TOTAL
    - FROM BOOK NATURAL JOIN USER_CART
    - WHERE USER_CART.ID = USER_ID;

3. Triggers

- **NEGATIVE_BEFORE_UPDATE**
  - This trigger is used before updating into the BOOK Table when the user tries to update a specific book and the quantity of a book if this update will cause the quantity of a book in stock to be negative.
  - In the trigger check if NEW.Copies be less than zero set **Error Message** that means the user needs to take the number of copies more than the number which is located in the BOOK Table for each book.
  - The message_text is 'The quantity of a book in stock can\'t be negative!'.
  - This is done through using the query:
    - IF (NEW.Copies < 0) THEN signal sqlstate '45000' set message_text = 'The quantity of a book in stock can\'t be negative!';
    - END IF ;
  - The number '45000' indicates Error occurs.

- **ORDER_AFTER_UPDATE**
  - This trigger is used after updating into the BOOK Table when the user updates a specific book and the quantity of a book drops from above a given threshold (the minimum quantity in stock) to below the given threshold.
  - In the trigger check if NEW.Copies be less than NEW.Threshold set **Warning Message** that means the number of Copies in a specific book be less than the threshold so must make a new Order.
  - The message_text is 'WARNING ! Minimum quantity in stock to below the given threshold!, Please order sufficient quantity'.
  - This is done through using the query:
    - IF (NEW.Copies < NEW.Threshold) THEN
    - signal sqlstate '01000' set message_text = 'WARNING! Minimum quantity in stock to below the given threshold!, Please order sufficient quantity';
    - END IF ;
  - The number "01000" indicates a warning occurs.

- **BEFORE_DELETE_ORDER**
  - This trigger is used before deleting into the ORDERS Table when the user deleting the order means that the order is received from the publisher.
  - In the trigger firstly declare two variables one to Copies, another to Threshold .
  - Set the values of them by execute select command from BOOK Table where the condition Where is that current ISBN = OLD.BOOK_ISBN which old indicates that before Delete execute.
  - In the trigger after that check if Re_Copies be less than Re_Threshold set **Error Message** that means the number of Copies in a specific book be less than the threshold so can't delete this Order.
  - The message_text is 'Can\'t delete the order before Confirming it!'.
  - This is done through using the query:
    - declare Re_Copies int; declare Re_Threshold int;
    - select Copies, Threshold INTO Re_Copies, Re_Threshold
    - from BOOK where ISBN = OLD.BOOK_ISBN;
    - IF (Re_Copies < Re_Threshold) THEN
    - signal sqlstate '45000' set message_text = 'Can\'t delete the order before Confirming it!';
    - END IF ;

- The number ‘45000’ indicates Error occurs.

- **INSERT_INTO_CART**
  - This trigger is used before inserting into the USER_CART table when the user tries to add a new book to its shopping cart so the procedure ADD_ITEM is called.
  - It’s used to check whether the available number of copies of a required book is enough for the required quantity in the user's request or not.
  - If it’s enough; then user’s process is done successfully and the new book is inserted into his shopping cart.
  - If it’s not enough; the process is cancelled and an error message appears containing the reason for cancellation.
  - This is done through using the query:
    - IF (NEW.QUANTITY > ALL(SELECT Copies FROM BOOK WHERE (ISBN = NEW.ISBN)))
    - THEN signal sqlstate '45000' set message_text = 'ERROR! required quantity is greater than currently available';
    - END IF ;

4. Indices

- **IN THE SEARCH FOR BOOKS**
  - Use indices to speed up searches.
  - By default the SQL makes the Primary key and the foriegn key index these types B TREE and unique .
  - So make another index to speed up the search which is to make an index on the category and then the Title of the book, because each book belongs to one category so can index the Title book under the Category Type.
  - This is done through using the query :
    - CREATE INDEX idx_Search
    - ON BOOK (Category, Title);

5. Transaction

- **CHECKOUT**
  - This transaction is used when the user's payment process starts execution after checking his credit card number and its expiry date.
  - This transaction mainly consists of 4 stages:
    - **Prepare temporary table:**
      - This table holds three attributes associated with each book belongs to the user shopping card that he is about to buy:
        - ISBN: the id of the book
        - QUANTITY: required number of book's copies
        - Copies - QUANTITY: this attribute represents the expected remaining number of copies associated with this book in BOOK table after completing the buying process successfully
      - This is done through using the query:
        - CREATE TEMPORARY TABLE TEMP
        - SELECT ISBN , QUANTITY, Copies - QUANTITY AS N
        - FROM USER_CART NATURAL JOIN BOOK
        - WHERE ID = USER_ID;
    - **Updating Book table with new number of copies:**
      - Just use the third column (N) in the temporary table to update the number of copies of sold books with their new value
      - If the updated value is negative then NEGATIVE_BEFORE_UPDATE trigger stops this process and marks it as invalid one
      - This is done through using the query:
        - UPDATE BOOK NATURAL JOIN TEMP
        - __SET BOOK.Copies = TEMP.N
        - __WHERE BOOK.ISBN > -1;
    - **Add to sold books:**
      - If previous two queries are completed successfully, then the books just bought by the user are added to the SOLD_BOOKS table to be used in reporting staff
      - This is done through using the query:
        - INSERT INTO SALED_BOOKS (BOOK_ISBN, customer_id)
        - SELECT ISBN, ID
        - FROM USER_CART
        - WHERE ID = USER_ID;
    - **Make the cart empty:**

- It's the last query in the transaction where the shopping cart of this user should be empty right now as he has completed the buying process
- In this stage, transaction just calls the predefined procedure REMOVE_ALL_ITEMS which is responsible for making the cart empty
- This is done through using the query:
    - CALL REMOVE_ALL_ITEMS(USER_ID);

6. Reports On Sales

- **Sales_for_month**

  This procedure is required to report the total sales for books in the previous month.
    - It retrieves all the books sold in the previous month, their prices and the number of copies which were sold for each book.
    - It uses YEAR() and MONTH() methods and the CURRENT_DATE to maintain the condition that the obtained books are ONLY for the previous month.

- **topCustomers**

  This procedure is required to report the top 5 customers who purchase the most purchase amount in descending order for the last three months.
    - It retrieves the names of these 5 customers with the total payment each customer paid.
    - It uses **GROUP BY Customer_ID.**
    - It also uses **ORDER BY** the total payment for each customer to achieve the required descending order

- **topBooks**

  This procedure is required to report the top 10 selling books for the last three months.
    - It retrieves the book_ISBN, Title and #sold copies for each book from these ten books.
    -
    - **GROUP BY BOOK_ID** and **ORDER BY #sold copies** are required to use here.

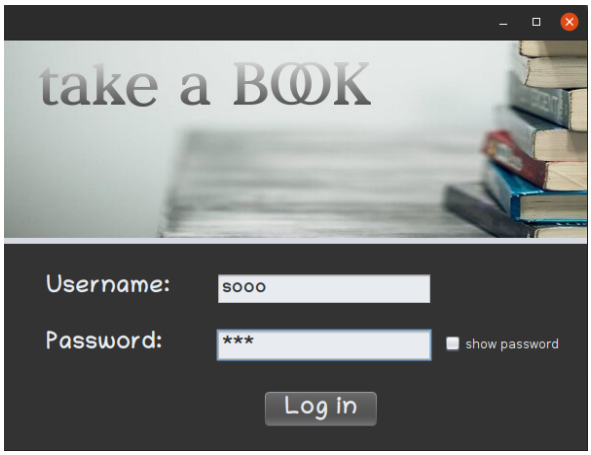
Java application

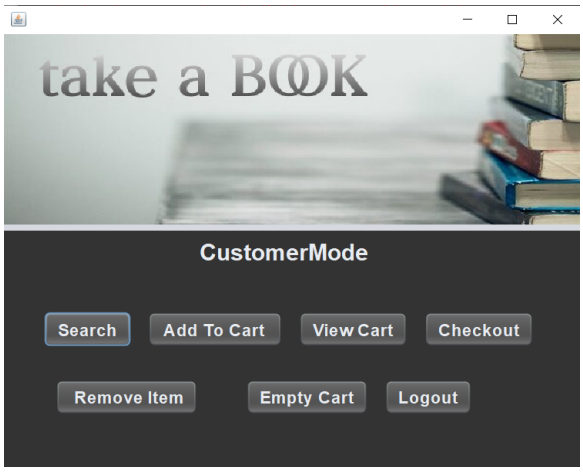
Main algorithm:

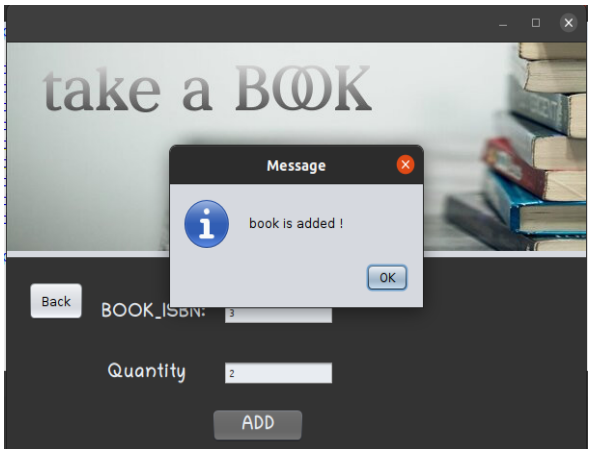
- This project is mainly divided into 2 packages; one manages the user's interface and the other manages the manager's interface.
- The starting frame is **login** form where the program checks whether the user is a valid user, a manager or invalid one
- Depending on previous decision it decides which form to be opened relative to the user privileges; normal form or high privilege form in case of managers or promoted users
- Each form consists of group of buttons and text fields to make the user able to perform his actions
- Clicking on any button takes one of two options:
    - Move the user to another form depending on the nature of the request
    - Call one of the database predefined procedures to communicate with the database
- In case of any error related to not-allowed query a message dialog appears with error description
- The user can move between forms using **back** button

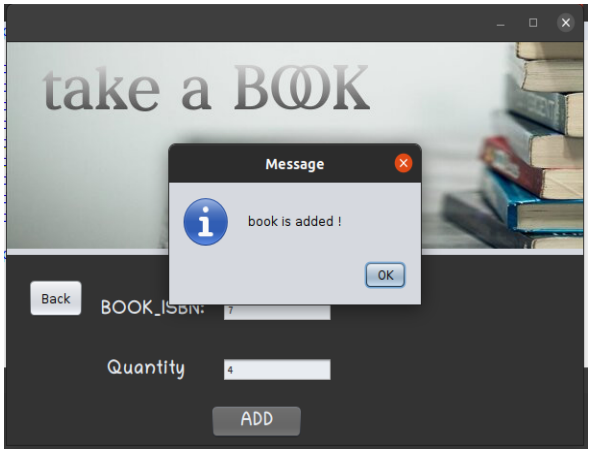
User package:

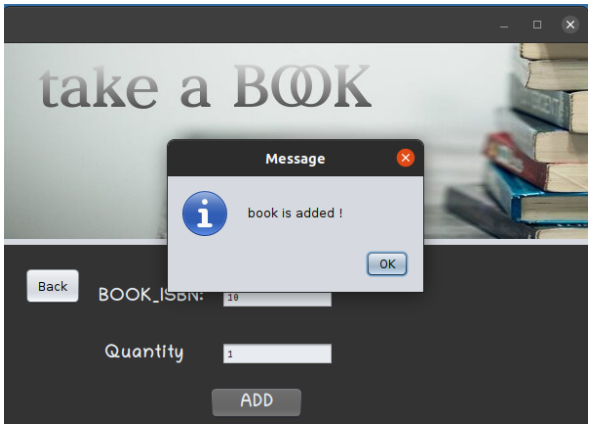
1. Add books to a shopping cart:

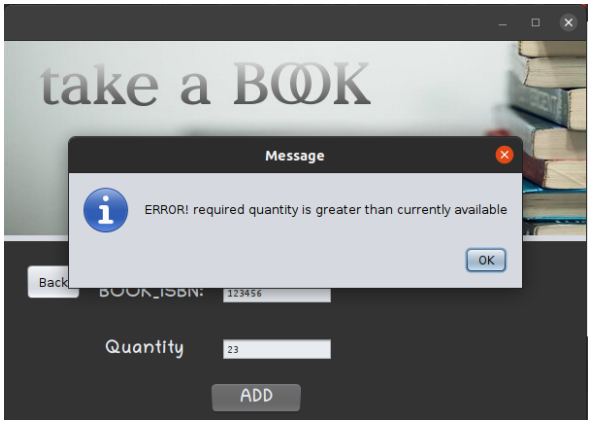




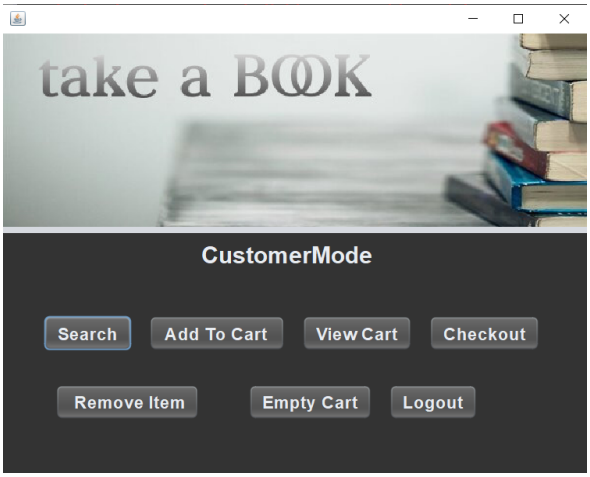


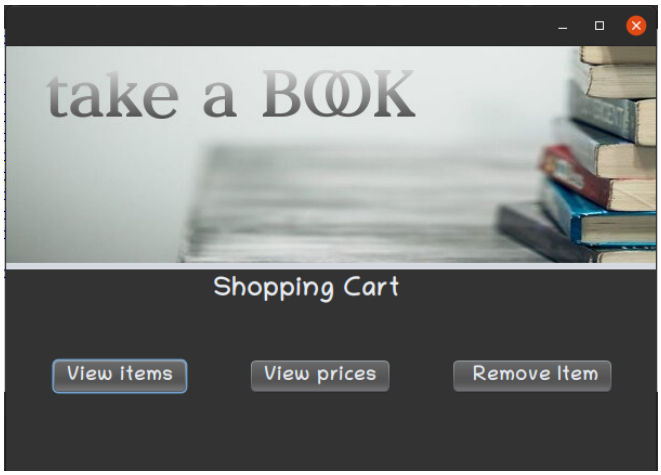




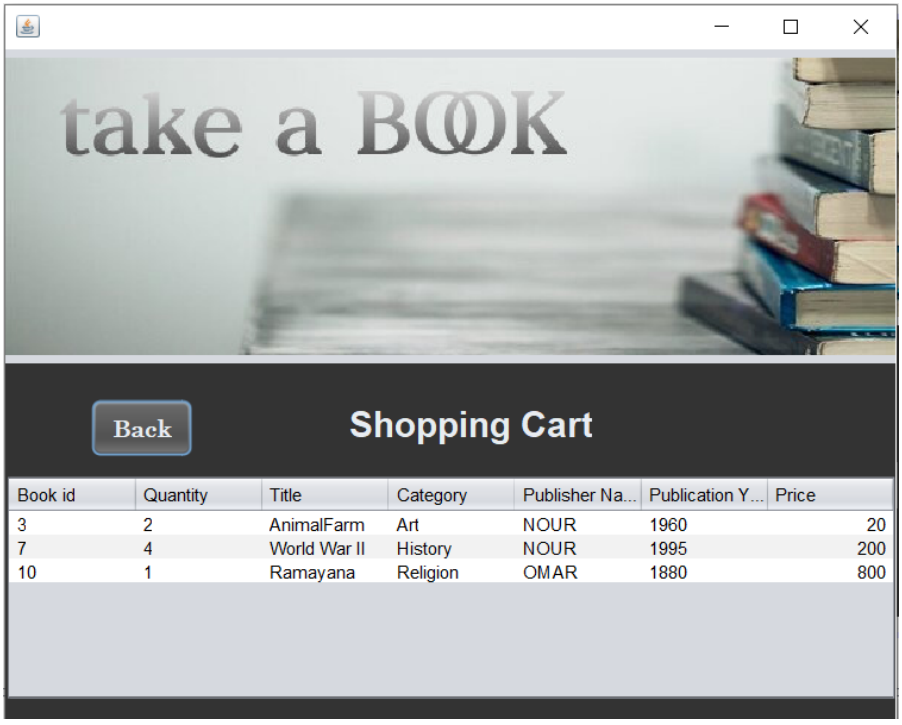


2. Manage the shopping cart:

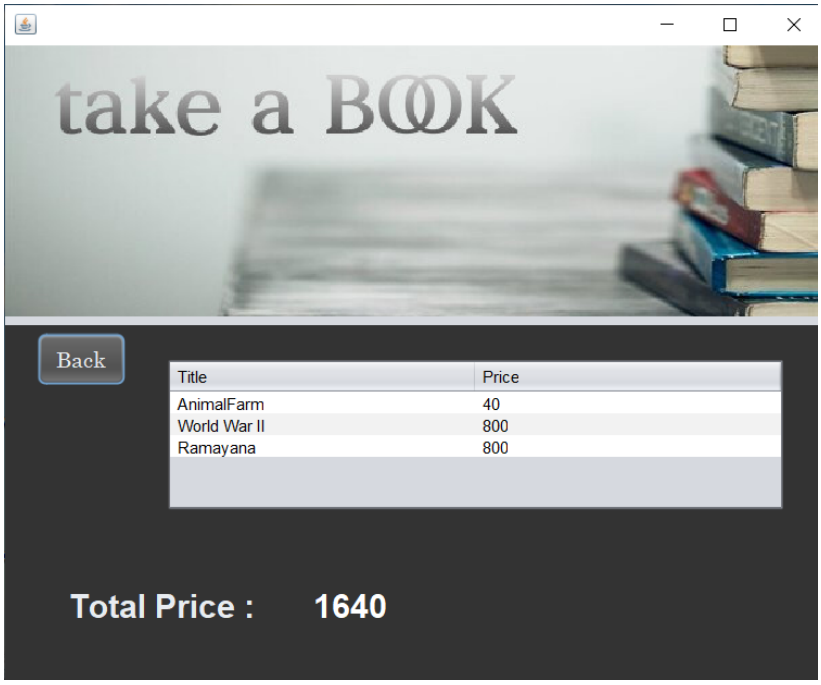




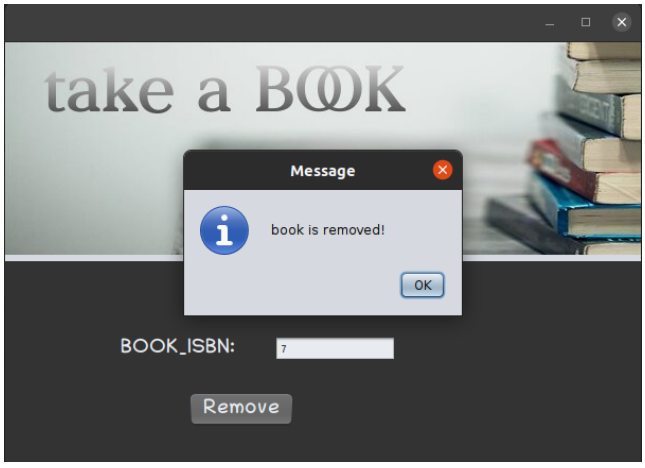
- **View the items in the cart**



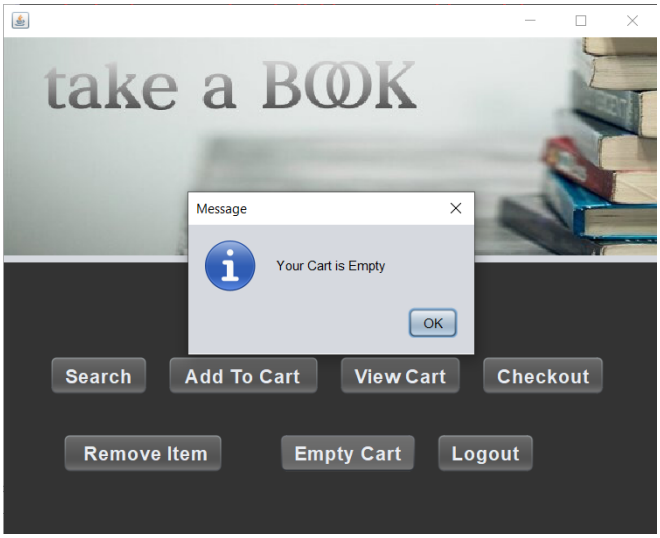
- **View the individual and total prices of the books in the cart**

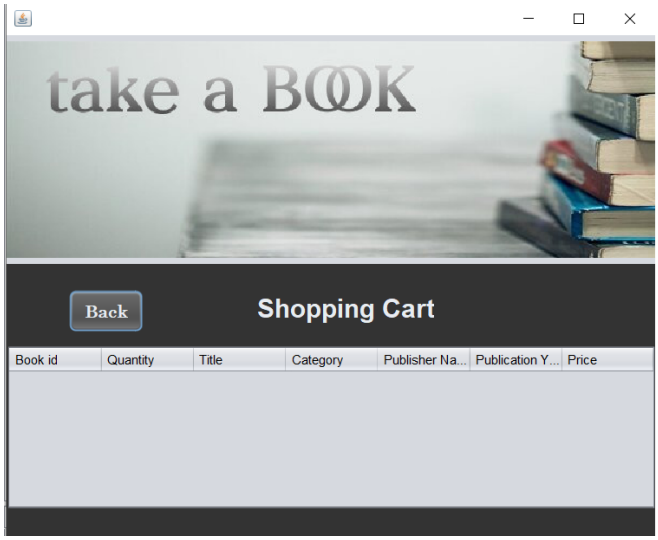


- **Remove items from the cart**

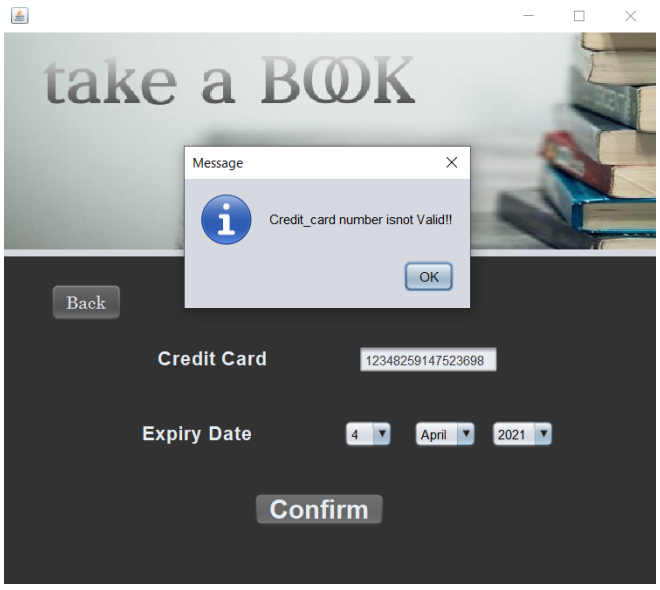


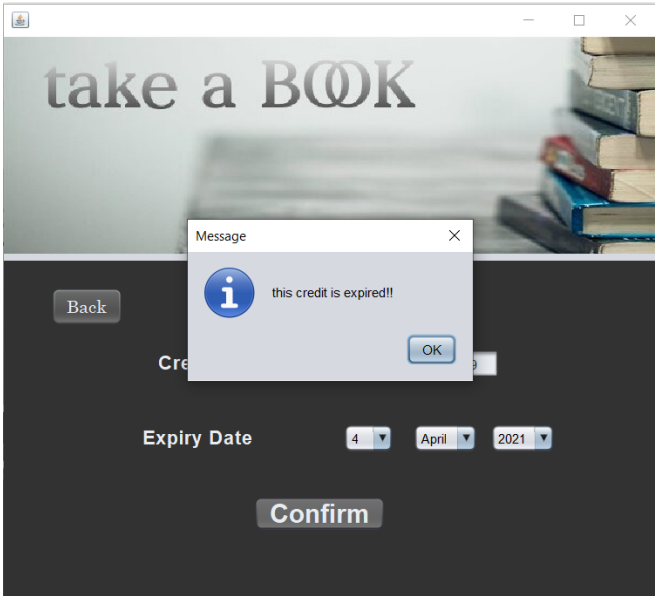
- **Remove all items from the cart**

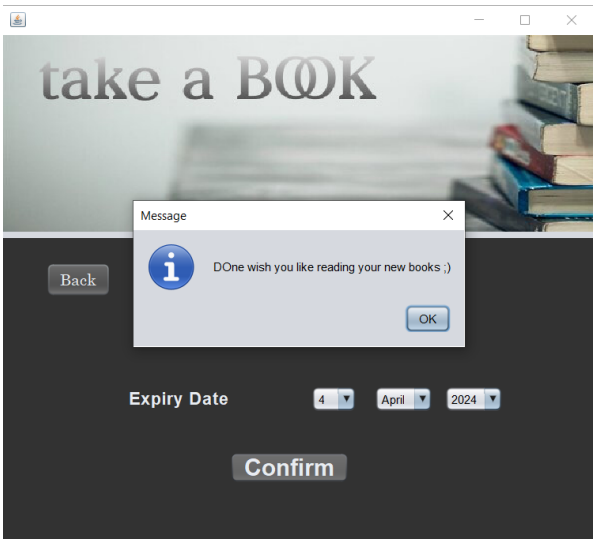


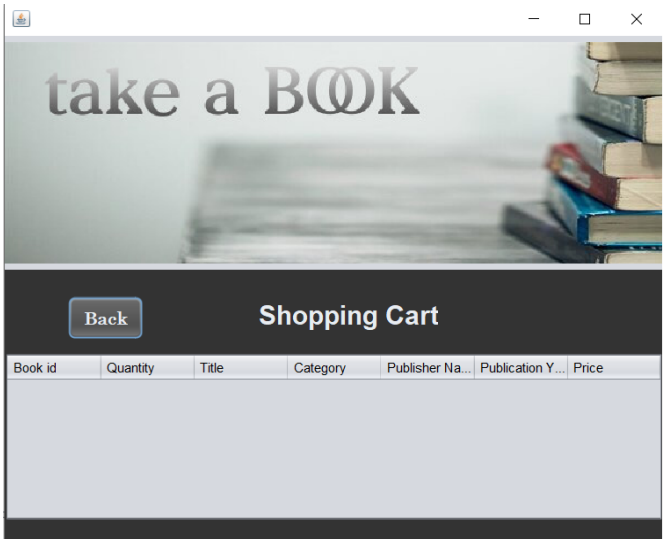


3. Checkout a shopping cart:

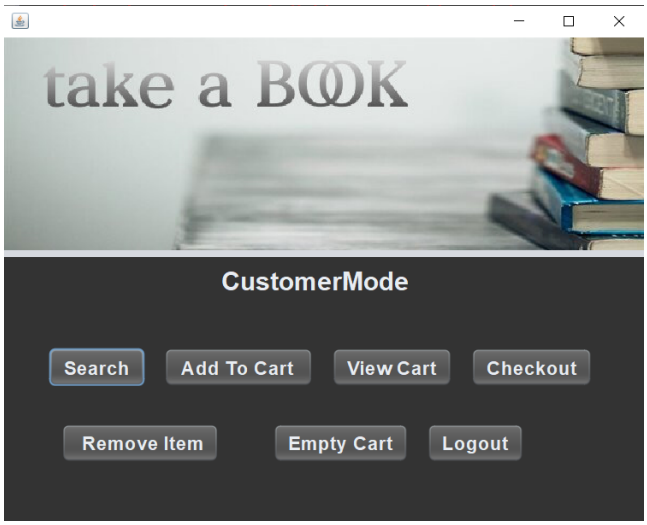


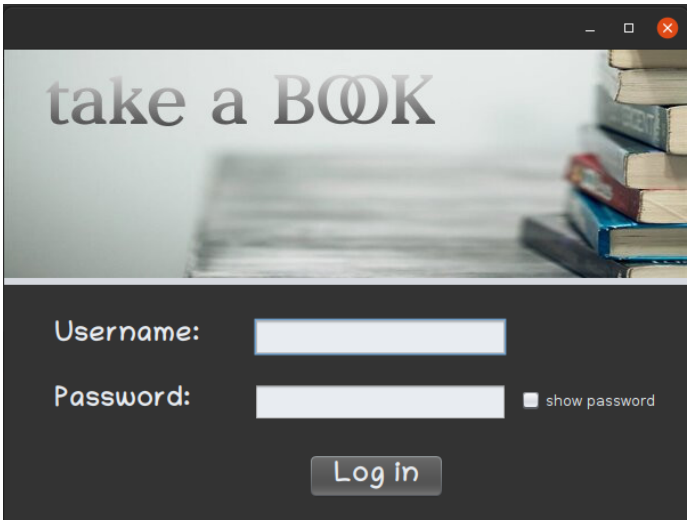






4. Logout of the system:

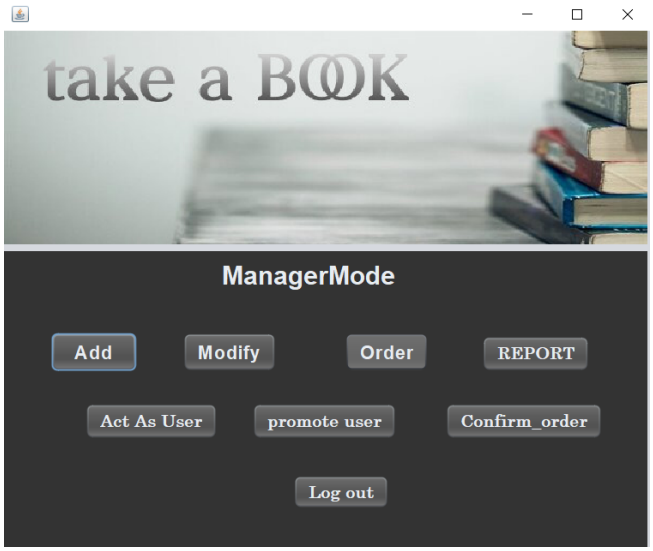




Manager package:

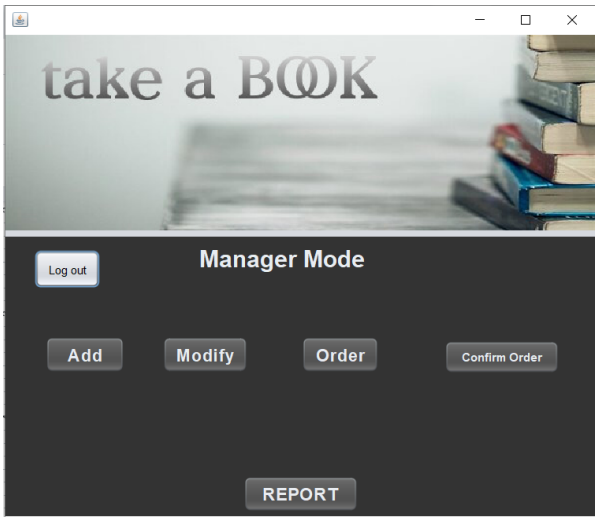
When a manager is logged in, the following window appears to him to let him choose what he wants to do.

★ If the manager wants to have user's options, he just need to press on "Act As User " button

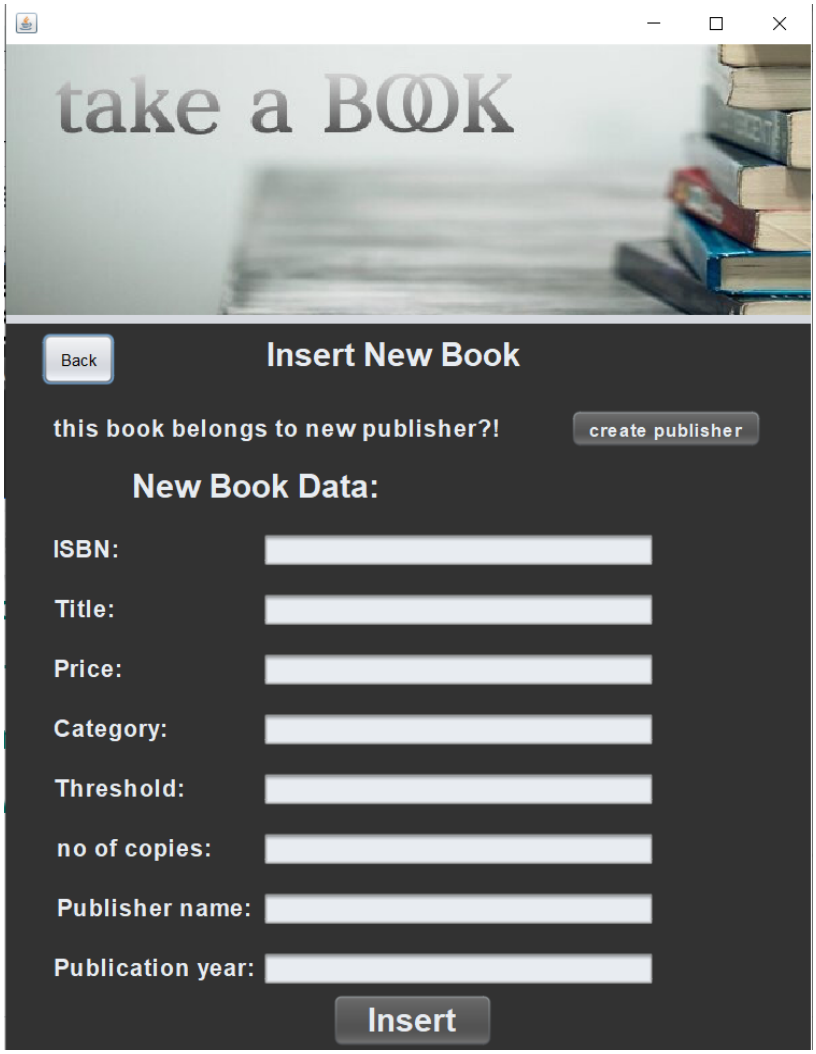


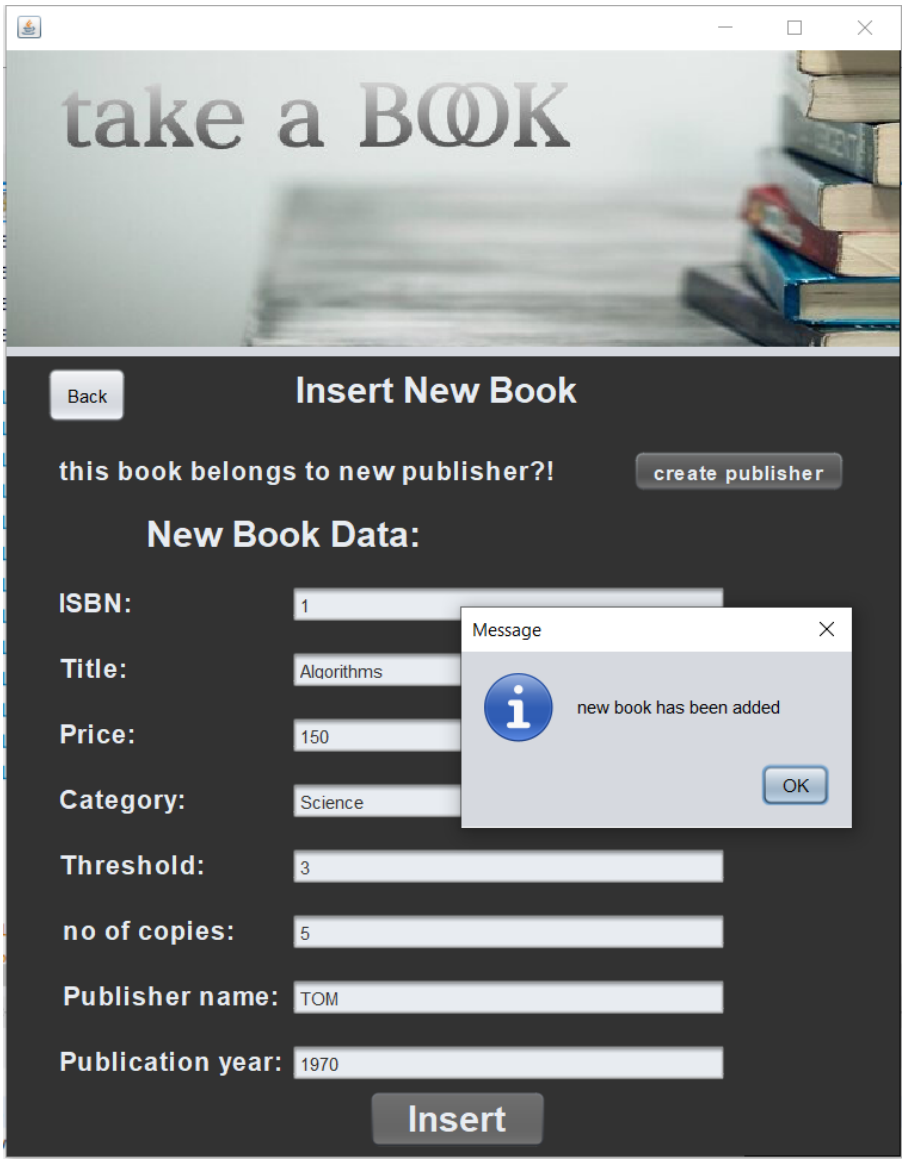
1. Add new books

→ To add new book to the bookstore, the manager pushes the "Add" button from the following window:



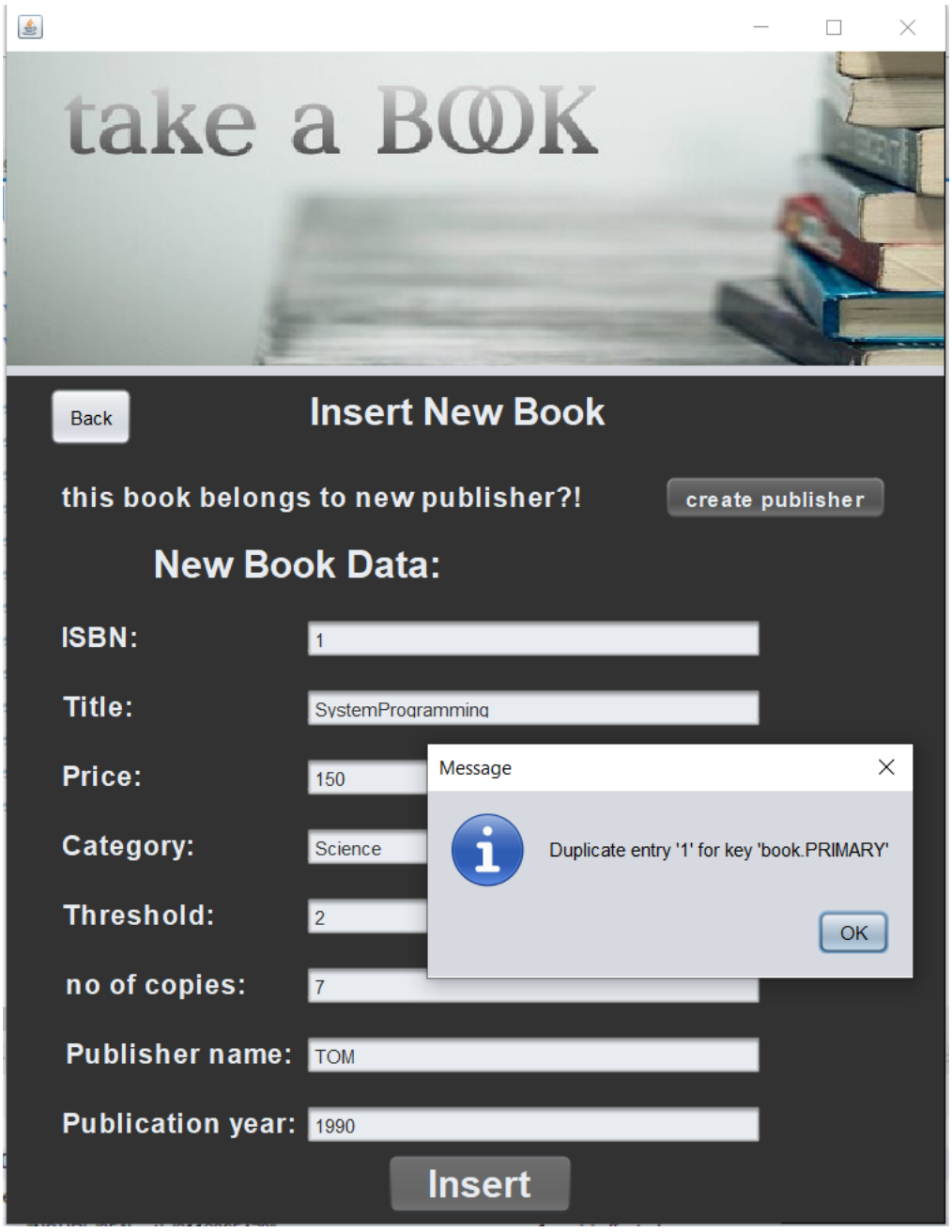
→ Then for the following window, he should insert books attributes, and then press “Insert” button:



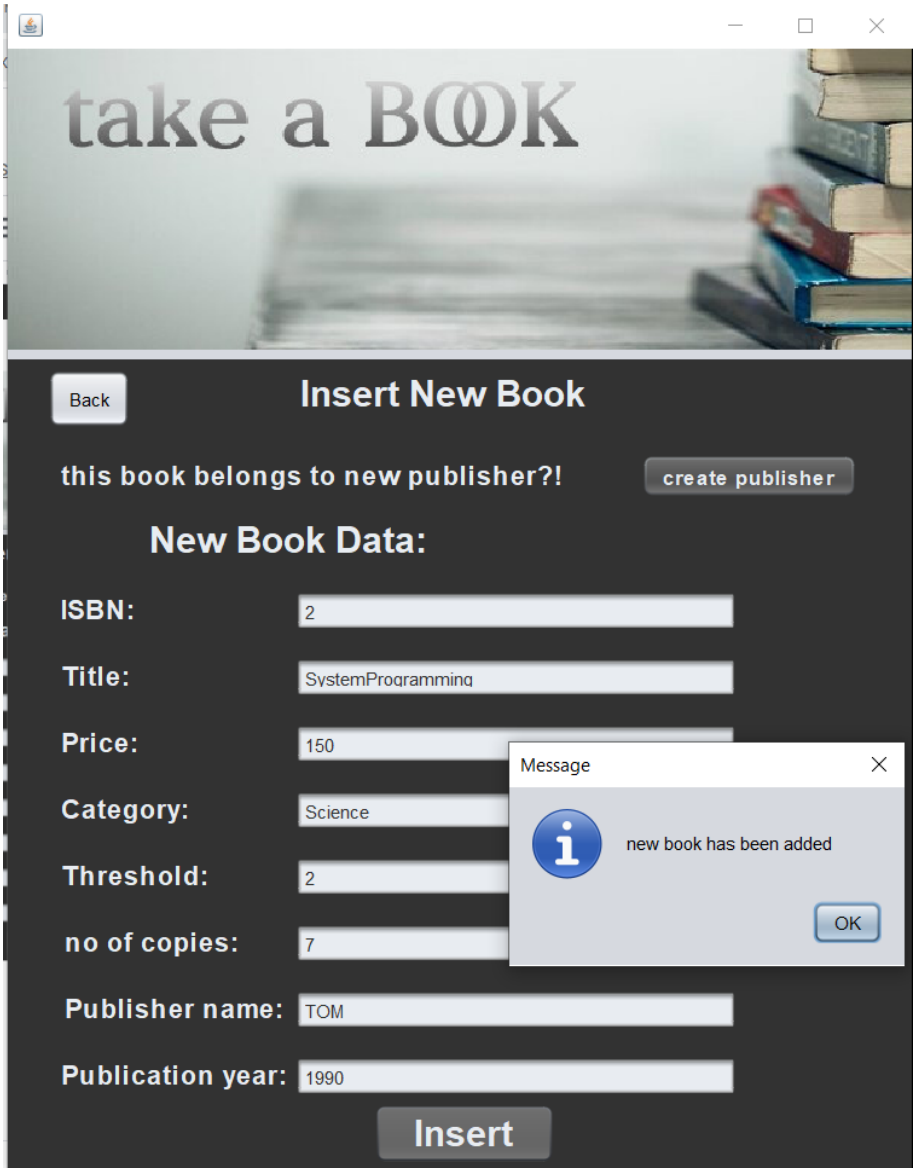


→ The Manger can back at any time to the Home page, By press on "Back" button:

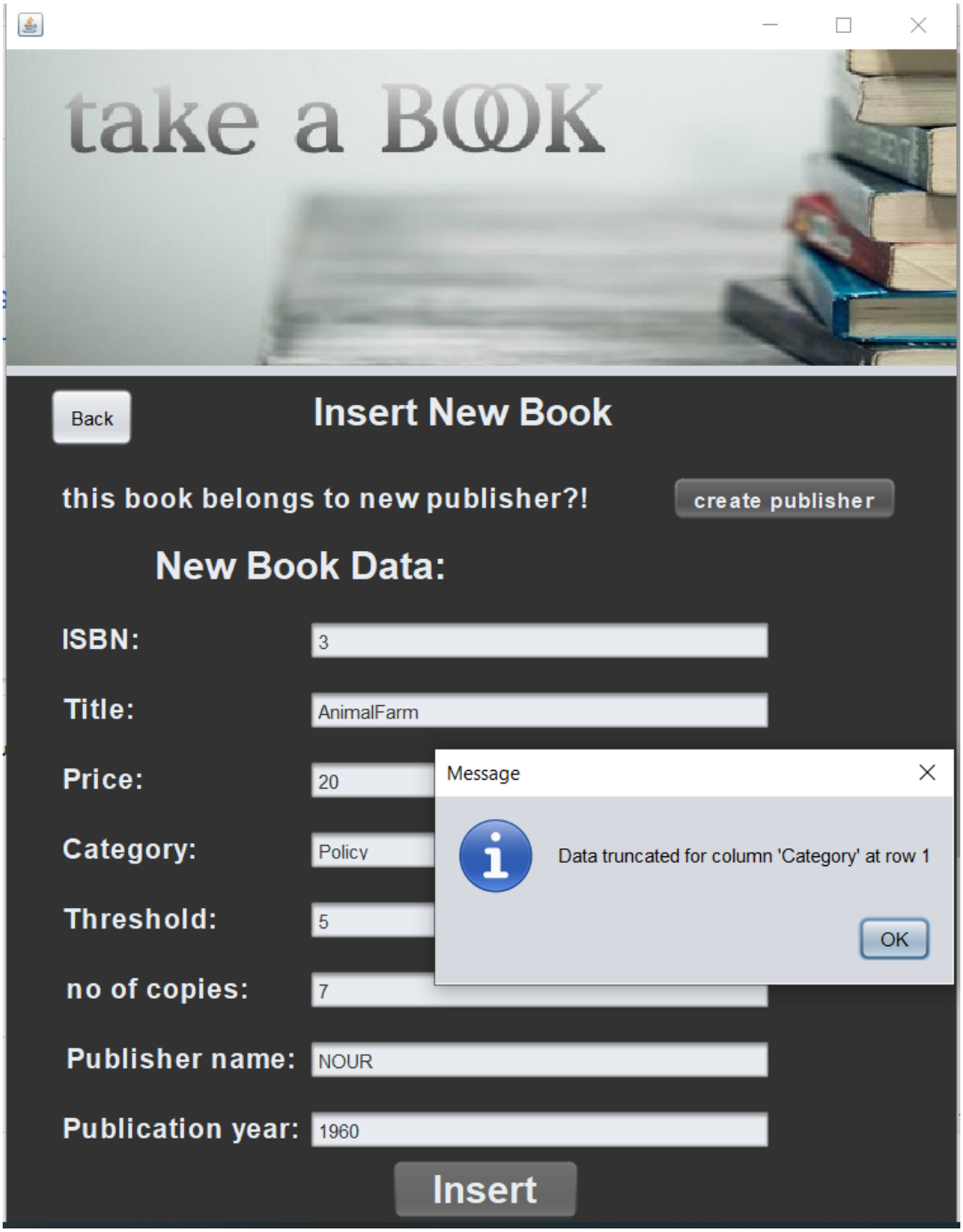
→ Then, if there is another book with the same “ISBN”, a message will appear to prevent this insertion.(Primary key constraint)



→ Another book to be added:

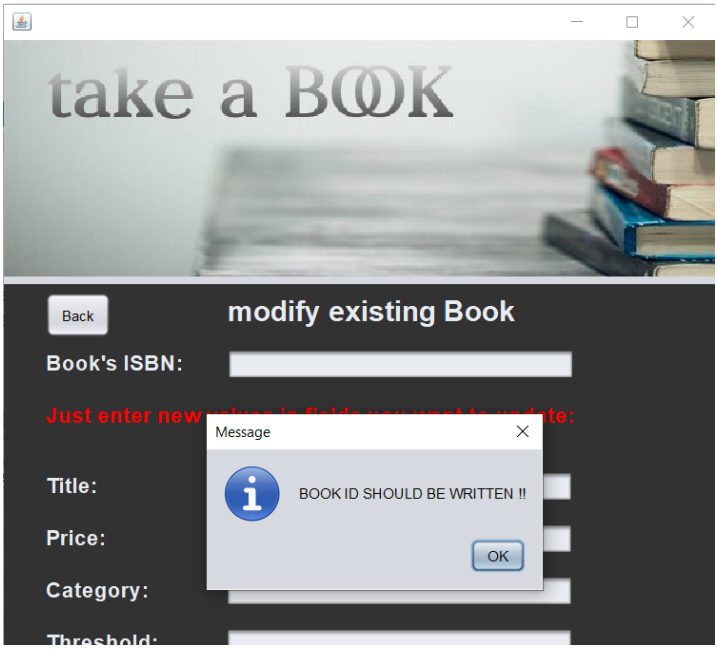


→ Another book to be add but with a category that isn't supported by this bookstore:

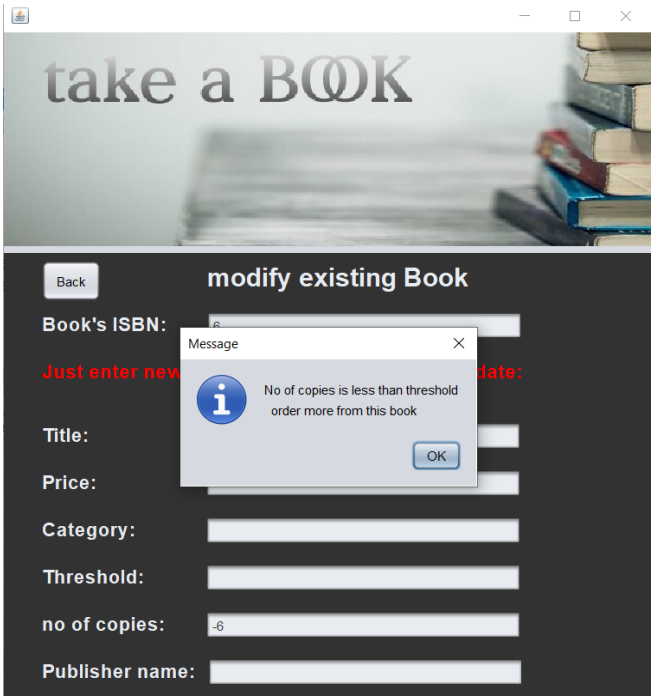


2. Modify existing books

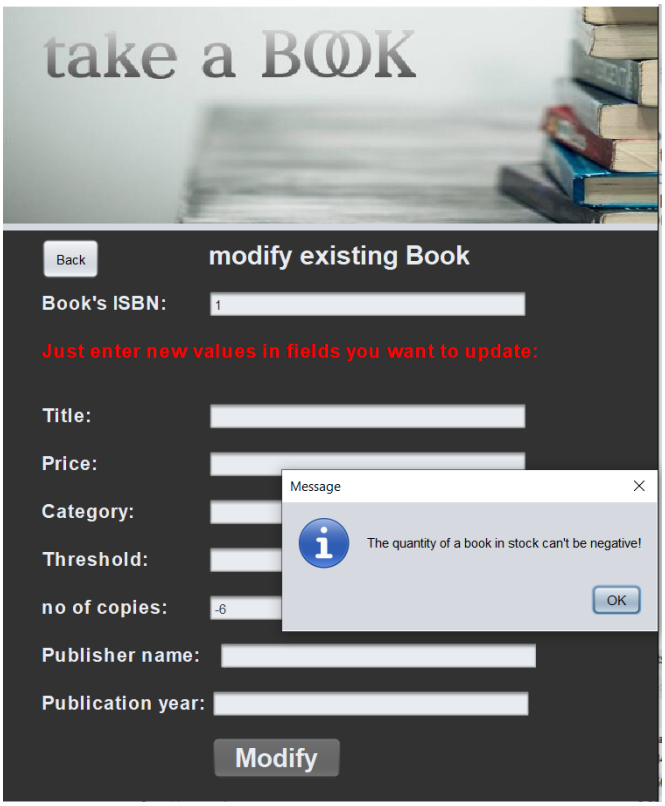
→ user should enter the id of book to be modified ：



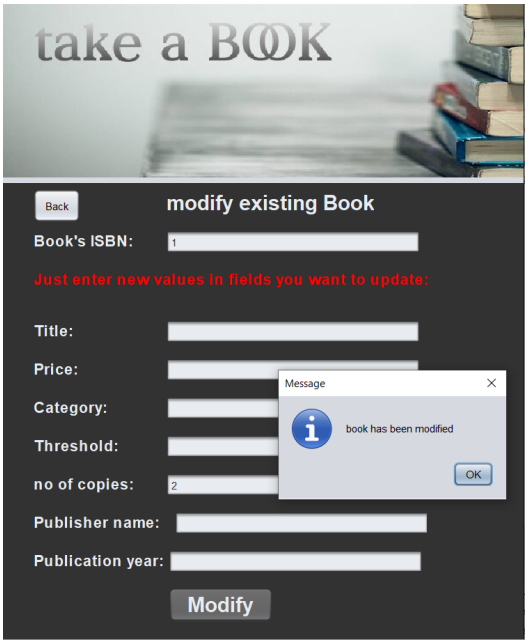
→if the manager modified book to no of copies less than the threshold warning should be appeared to order more from this book ：



→ when try to update number of copies to be less than zero Error Message appear :



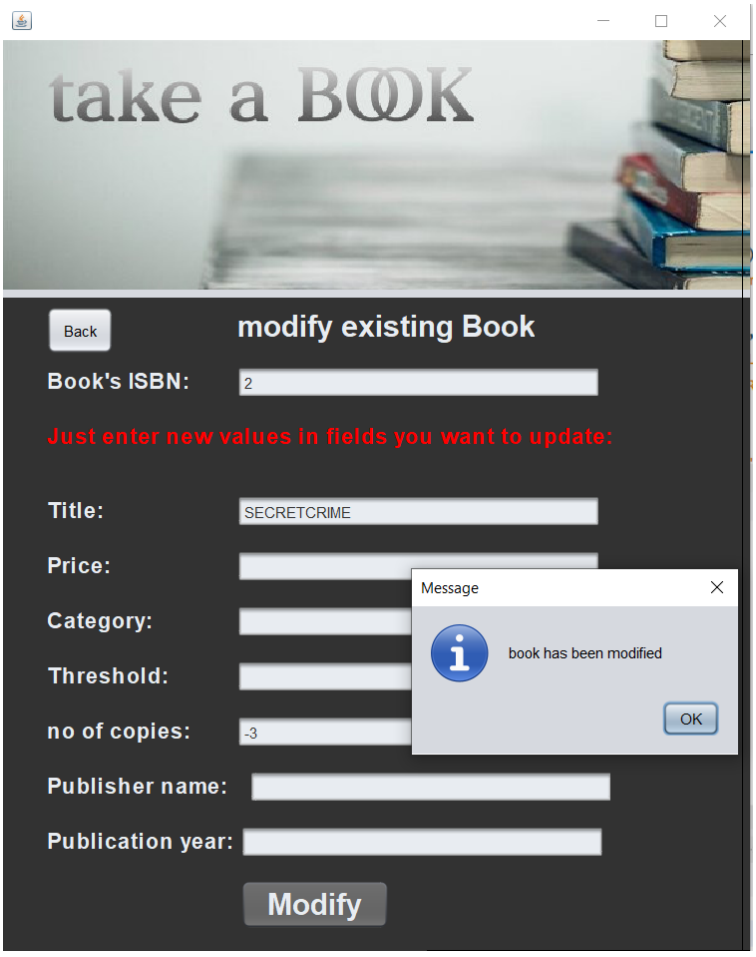
→ In the normal case update add 2 copies in the book id 1 :



→ the result of BOOK Table as shown after Modify :

ISBN	Title	Publication_Year	Price	Category	Threshold	PUBLISHER_Name	Copies
1	Algorithms	1970	150	Science	3	TOM	7
2	SystemProgramming	1990	150	Science	2	TOM	7

→ Another the normal case update sold 3 copies and change the Title of the Book to "SECRET CRIME" in the book id 2 :

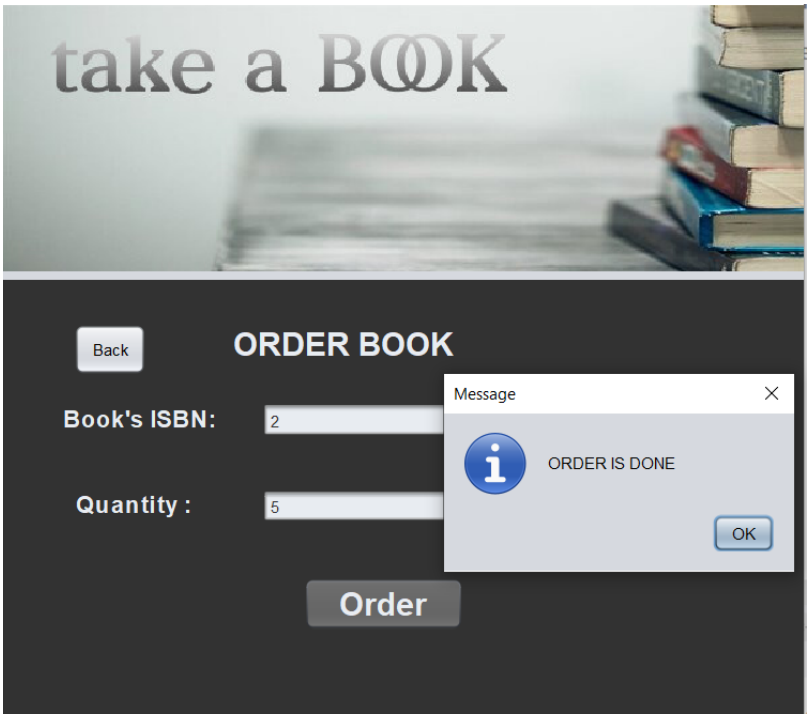


→ The result of BOOK Table as shown after Modify:

ISBN	Title	Publication_Year	Price	Category	Threshold	PUBLISHER_Name	Copies
1	Algorithms	1970	150	Science	3	TOM	7
2	SECRETCRIME	1990	150	Science	2	TOM	4

3. Place orders for books

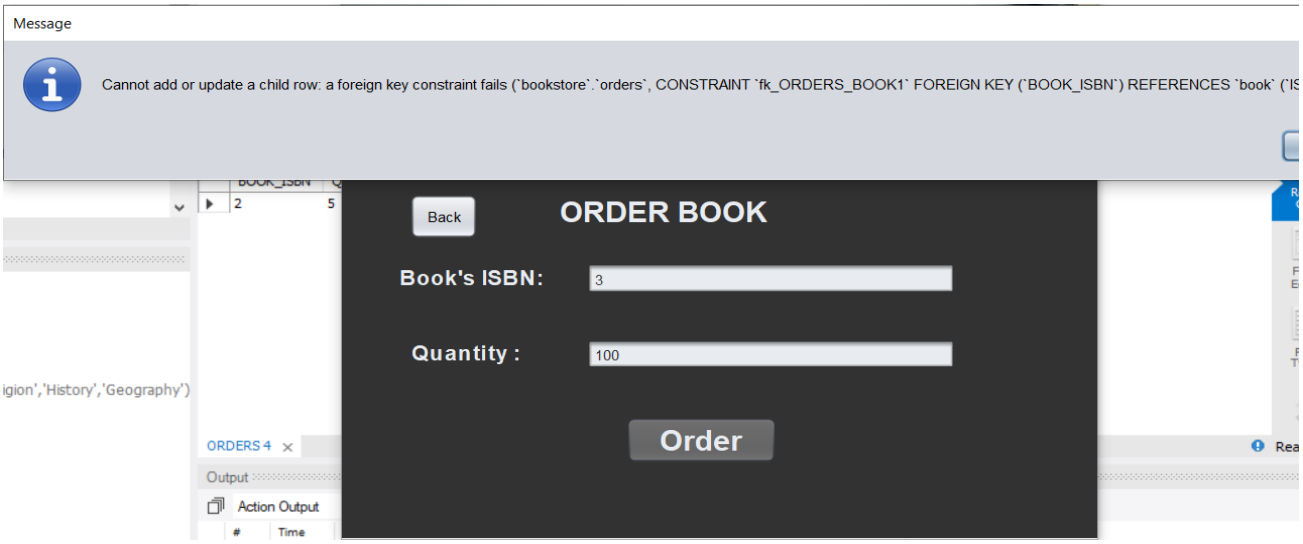
→ when placing a normal Order.



→ The result of ORDERS Table as shown after insert order:

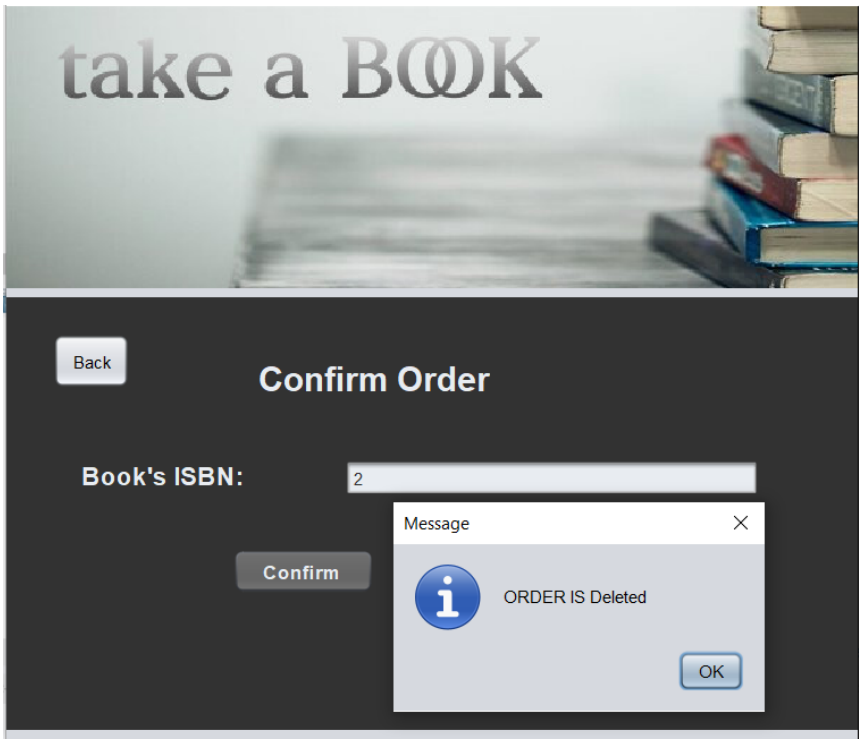
BOOK_ISBN	Quantity
2	5

→ The result of ORDERS Table as shown after wrong order which number ISBN NOT exist:



4. Confirm orders

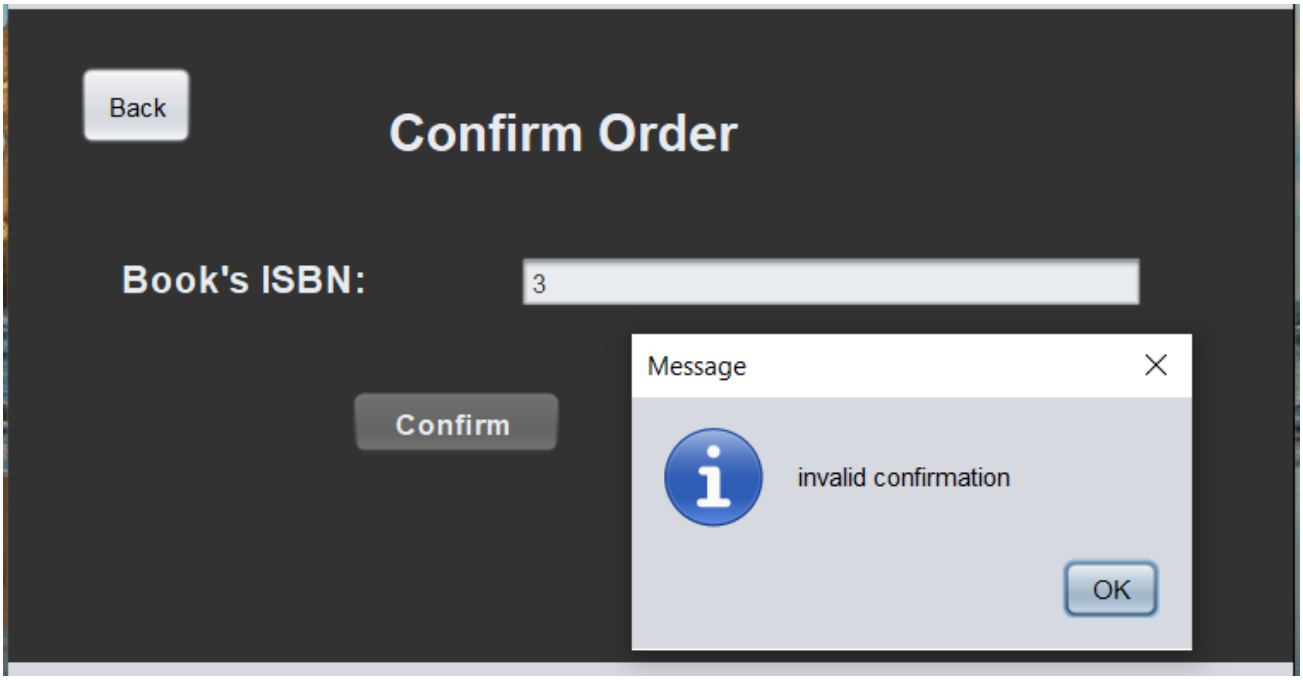
To confirm that the order had been delivered to the bookstore, choose confirm then insert the ISBN of the book's order.
Then, this order will be deleted from the orders table.



Here is the order's table after confirming the order:

	BOOK_ISBN	Quantity

→ when trying to delete an order that isn't in the order table, an error message will appear.

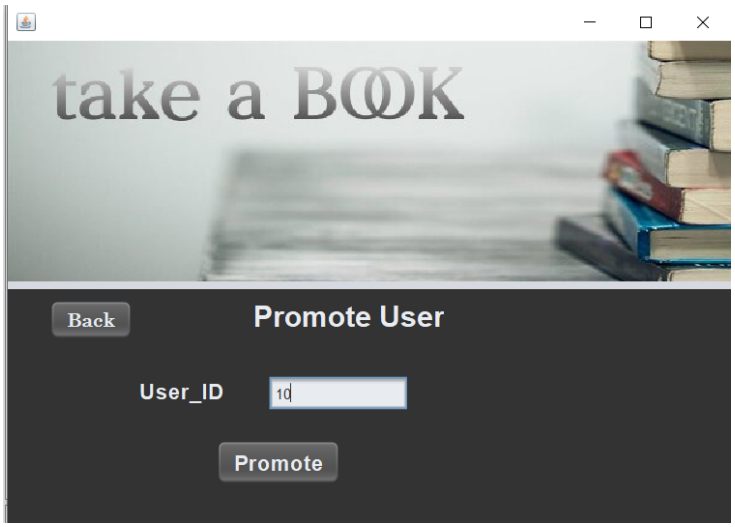


5. Promote registered customers to have managers credentials

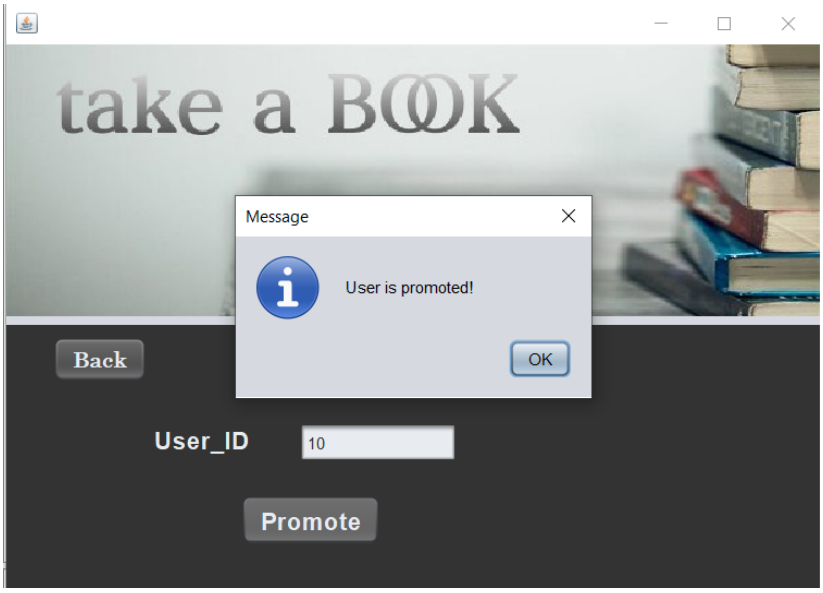
Firstly, the user is registered as an ordinary user with his specific privileges.

```
insert into USERS values ('10','enas', 'morsy', 'kjh', '0123456', 'mandara', 'enas', 'emms99',false);
```

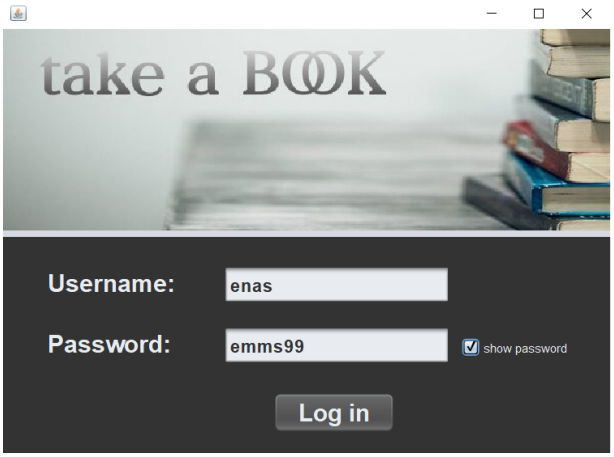
Then the manager decides to promote this user by its ID

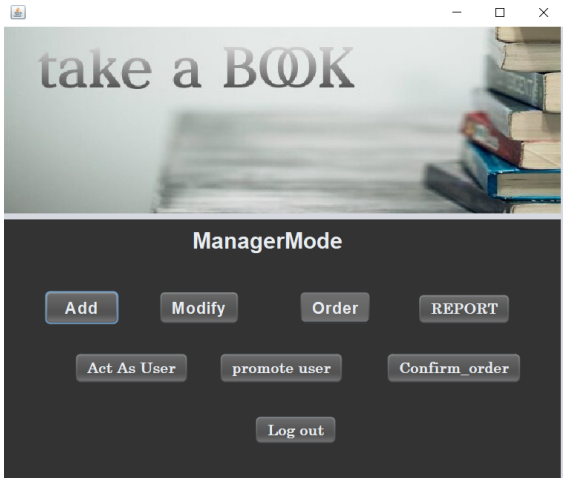


Then, the user is promoted:



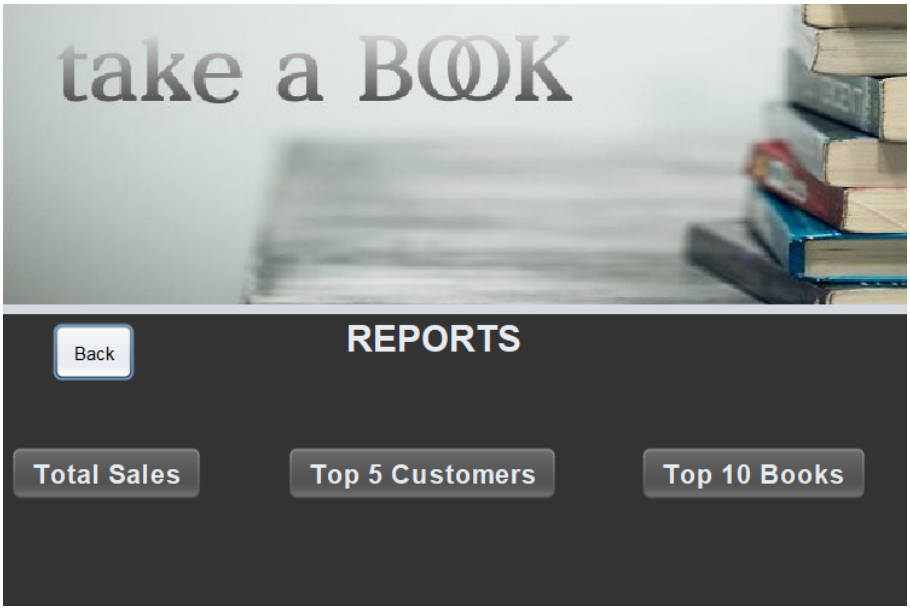
So, when this user tries to log in again, the operations of the managers are now allowed to this user.



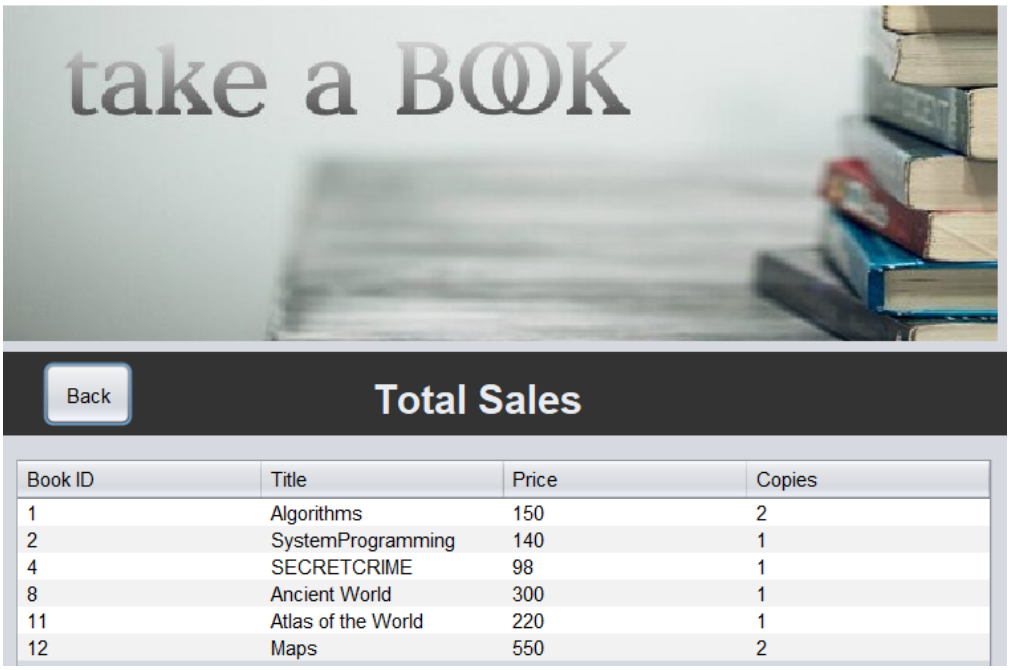


6. View reports on sales

→ To view reports, the user has to push the "Reports" button from the Manger mode window, then choose which report he needs from this window:

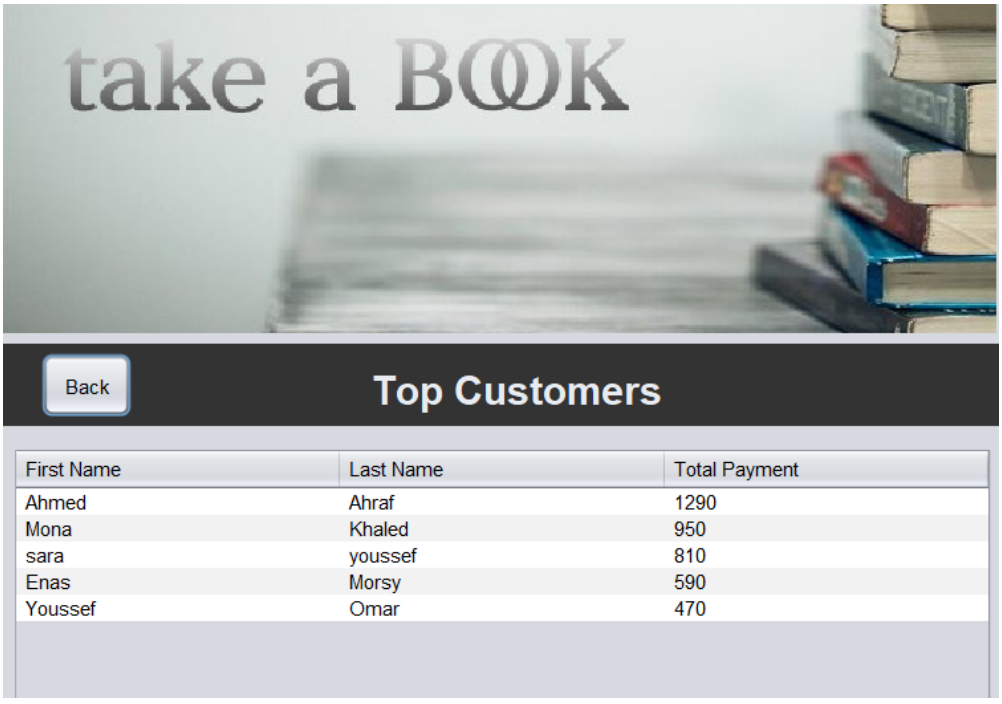


- For **Sales_for_month,** push the **Total Sales** button.
  Here is the report of the inserted database



Book ID	Title	Price	Copies
1	Algorithms	150	2
2	SystemProgramming	140	1
4	SECRETCRIME	98	1
8	Ancient World	300	1
11	Atlas of the World	220	1
12	Maps	550	2

- For the **top 5 Customers,** push the **Top 5 Customers** button.
  Here is the report of the inserted database



First Name	Last Name	Total Payment
Ahmed	Ahraf	1290
Mona	Khaled	950
sara	youssef	810
Enas	Morsy	590
Youssef	Omar	470

- For the **Top 10 BOOKS,** push the **Top Books** button.
  Here is the report of the inserted database