**COMP3005 Project Report**

Tom Lam – 101114541

**Assumptions:**

Cardinality:

* Participation type is indicated by the number of participants next to each entity in the ER diagram. Specifically, “1..\*” or “1” means total participation while “0..\*” or “0..1” means partial participation
* A book can be published by exactly 1 publisher, A publisher can publish 1 to many books
* A book can be collected by 0 to 1 owner, an owner can collect 0 to many book. There’s no book that belongs to 2 different owner
* A book can be ordered by 0 to many orders, an order can order 1 to many book
* An order can be placed by exactly 1 customer, a customer can place 0 to many order

Other:

* Different authors have different name (to avoid the use of author\_id)
* Different publishes have different name (to avoid the use of publisher\_id)
* Different customer has different name (to avoid the use of customer\_id)

**ER Diagram**

Diagram

Description automatically generated

**Relational Schema**

Book(ISBN, publisher\_name, book\_name, genre, description, num\_of\_pages, price)

Author(ISBN, name)

Publisher(name, email, bank\_account, balance, address, phone\_number)

Owner(name, bank\_account, balance, address, email, phone\_number)

Collect(ISBN, owner\_name, unit\_in\_stock, unit\_sold, revenue, expense, profit, publisher\_split)

Order(order\_id, billing\_address, shipping\_address, status, ordered\_date, estimated\_arrivate, location)

Customer(name, bank\_account, balance, billing\_address, shipping\_address)

CustomerOrder(order\_id, customer\_name)

OrderBook(ISBN, order\_id, unit\_ordered)

**Normalization**

Functional dependencies

Book(ISBN, publisher\_name, book\_name, genre, description, num\_of\_pages, price)

ISBN 🡪 name, publisher\_name, book\_name, genre, description, num\_of\_pages, price

Author(ISBN, name)

ISBN, name 🡪 ISBN, name

Publisher(name, email, bank\_account, balance, address, phone\_number)

name 🡪 email, bank\_account, balance, address, email, phone\_number

Owner(name, bank\_account, balance, address, email, phone\_number)

name 🡪 bank\_account, balance, address, email, phone\_number

Collect(ISBN, owner\_name, unit\_in\_stock, unit\_sold, revenue, expense, publisher\_split)

ISBN 🡪 owner\_name, unit\_in\_stock, unit\_sold, revenue, expense, publisher\_split

Order(order\_id, billing\_address, shipping\_address, status, ordered\_date, estimated\_arrivate, location)

order\_id 🡪 billing\_address, shipping\_address, status, ordered\_date, estimated\_arrivate, location

Customer(name, bank\_account, balance, billing\_address, shipping\_address)

name 🡪 bank\_account, balance, billing\_address, shipping\_address

CustomerOrder(order\_id, customer\_name)

order\_id 🡪 customer\_nameßœ

OrderBook(ISBN, order\_id, unit\_ordered)

ISBN, order\_id 🡪 unit\_ordered

All relations are already in BCNF because their functional dependency is also the superkey

**Schema Diagram**

Graphical user interface, application, Word

Description automatically generated

**Implementation:**

***Architectures:***

The table creation “table.sql” and insertion “insertion.sql” files can be found in the “database\_initialize” directory.

The program entry point is in Main.java. It initializes the Application and run it.

The functionalities are implemented by 3 main classes:

* View.java handles all interaction between the store and its users (Customer or Owner)
* JDBCController.java handles all interaction with the store and the database. This is where all database querries and updates are implemented
* Application.java implements all application requirements stated in the **“Problem Statement”** of Look-Inna-Book by using the View and JDBCController classes to interact with the store users with the store databases

The remainings are helper classes of which main functionalities are storage and display (toString)

* Customer.java stores informations about a customer
* Owner.java stores information about an owner
* Book.java stores temporary book information got from the database
* BookOrder.java stores a Customer’s chosen Book with the number of that book that’s going to be ordered when checkout.
* Basket.java stores Customer’s BookOrder(s) before checkout. Basket will be emptied after checkout or restart
* Order.java stores an order information when tracking an order

***Program setup:***

1. Create the database and initialize it by running everything inside “table.sql” and “insertion.sql”.
2. In Main.java. Please change the following paramenters  
    PORT: database port  
    DATABASE\_NAME: name of created database  
    USERNAME: user name to log into database  
    PASSWORD: password of the username  
   to the appropriate values so that JDBCController can connects to the database

***Interfaces:***

*Different views:*

When first started, you will be greeted with the Main view and required to choose either Customer or Owner view:

A picture containing text

Description automatically generated

Customer view (not logged in):

Text, table

Description automatically generated with medium confidence

Owner view (not logged in):

Table

Description automatically generated

Obviously, you are allow to switch between view by exiting back to the Main view.

*Customer View:*

When not logged in, the customer can browse books, add book to basket, remove book from basket, show basket, and log in.

Text, table

Description automatically generated with medium confidence

When logged in, the customer gains the ability to checkout the basket, track order, and log out. And of course, loses the ability to log in.

Text

Description automatically generated

Browsing books, adding and removing to/from basket works the same regardless of whether the user is logged in or not. More over, when logging in or logging out, the basket is not reseted.

*Customer Browse Book:*

When choose to browse book, you will be brought to the Browse Book view. Which gives to option to search using the filters, show, clear, and modifies the filters.

Text

Description automatically generated

Initially, all filters are empty:

Text

Description automatically generated

Add to filters to search for books that are writen by Author3 and published by either Publisher2 or Publisher1:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Resulting filters:

Text

Description automatically generated with medium confidence

Resulting search, the query is also printed for visualization purpose:

Text

Description automatically generated

Customer add book to basket (after going back to the main customer view):

Choose the ISBN from the list printed and the quantity to add a book order to basket.

Add 10 books with ISBN 5

A picture containing table

Description automatically generated

Add 10 books with ISBN 5:

Table

Description automatically generated with medium confidence

The basket information after adding:

Text

Description automatically generated