|  |  |  |
| --- | --- | --- |
| UnitecHorizontalLogo | ISCG 5421 Programming Principles and Practice  **Assignment**  **Library Management System** | |
| Sherman Perry ID 1441589  Nataliya Cardillo ID 1443595  Christian Hayward ID 1454662  Aubrey Fernandes |  |  |

**Contents**

Introduction

Overview: requirements (version of scenario and specifications)

Flowcharts: User Types, Calculating Fine

UML Use Case Diagram

UML Sequence Diagrams

SQL diagrams

Database Tables

Class diagrams: GUI Classes, Constructor Classes, DAO Classes

Graphic User Interface Screens

Agendas and Minutes

**Introduction**

The project Library Management System is a library management software for monitoring and controlling basic operations such as adding new customer, new books, updating information, searching books and customers, borrowing and returning books.

It is developed in Java.

It features a familiar user interface, combined with searching options and reporting capabilities. The report generation option makes it possible to generate reports based on different filters.

Our Library Management System has next modules:

* Insertion to Database Module
* Extracting from Database module
* Report Generation module
* Search Facility system

**Overview: requirements**

**Receptionist -- Has permission to:**

* Register customer
* Search customer table
* Enter the name
* Enter address
* Enter contact number
* Change the status of borrowed books to returned books

**Customer – Has permission to:**

* Search books by title category, author, keyword, publisher
* Borrow books

**Administrator – Has permission to:**

* Search books by title topic, author publisher
* Add book from system
* Delete book from system
* Modify/Update the details of the book from system
* Add supplier to database
* Update supplier to database
* Delete supplier to database
* Add book to database
* Delete book to database
* Create username and password for the receptionist.

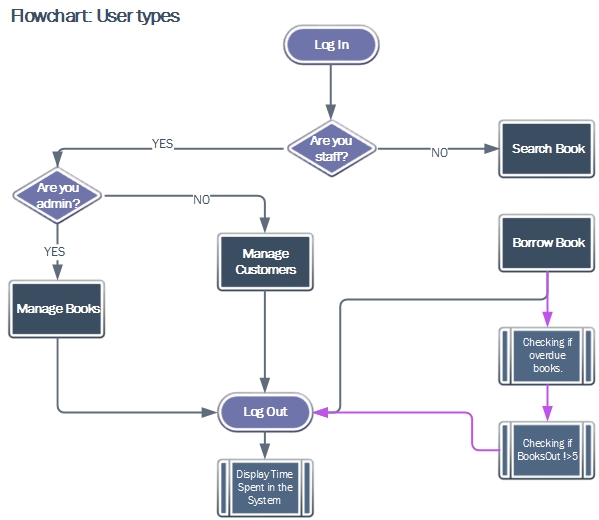
**System: functionality**

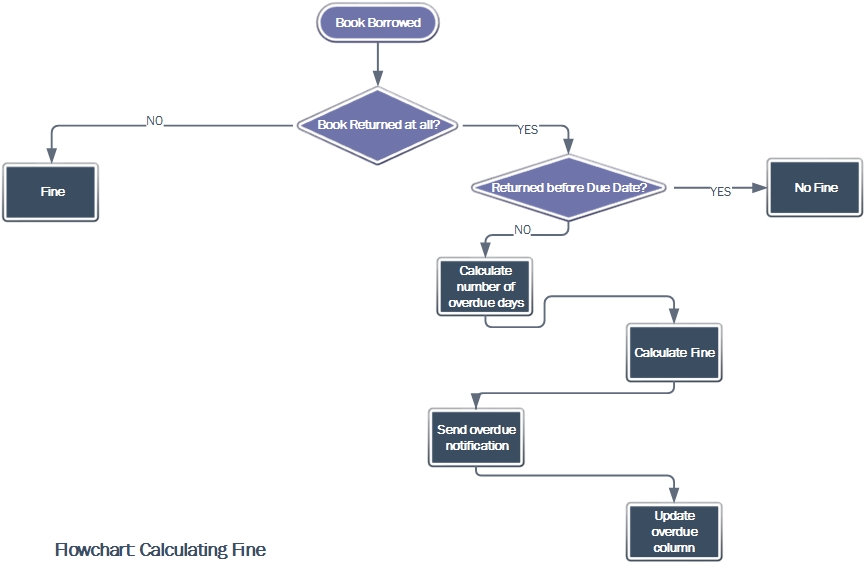
* Track Customers’ entry and exit time to the library system
* Restrict the amount of books a customer can borrow (max 5)
* Track length that customer is borrowing book for
* Fine 60cents per day that book is borrowed past due date
* Notification sent to customer about the fine
* If customer lose book a fine equals book’s cost will be added

**Report generating**

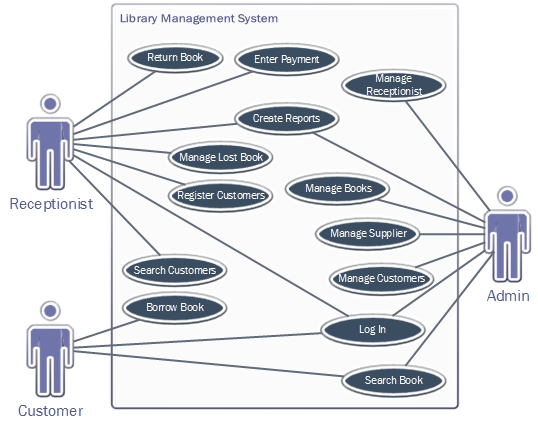
* Book lending report (Daily, Monthly, Annually),
* fine report (book lost, late return with fine),
* book purchased report
* Optional extra reports-- Popular books report Resources Collection Report

**Flowcharts**



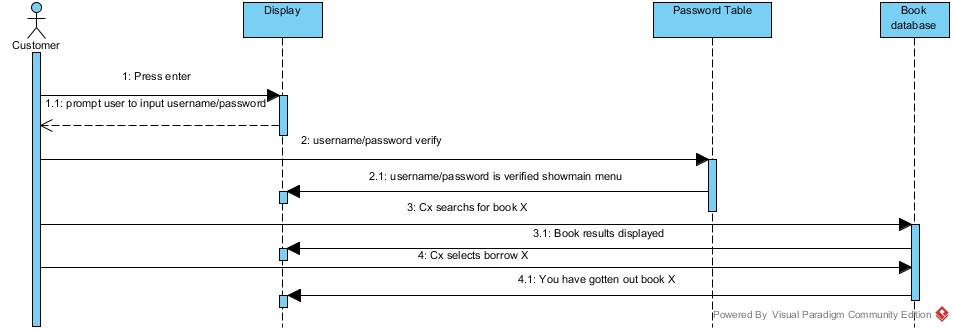


**UML Use Case Diagram**



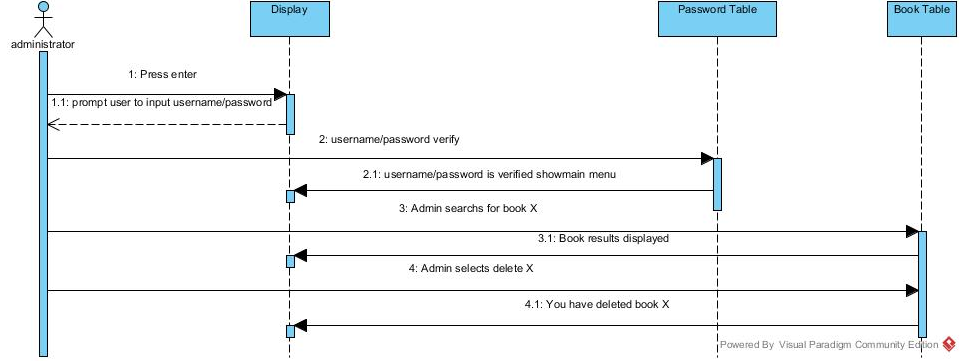
**UML Sequence Diagrams**

**Customer**

Example of customer searching and borrowing book

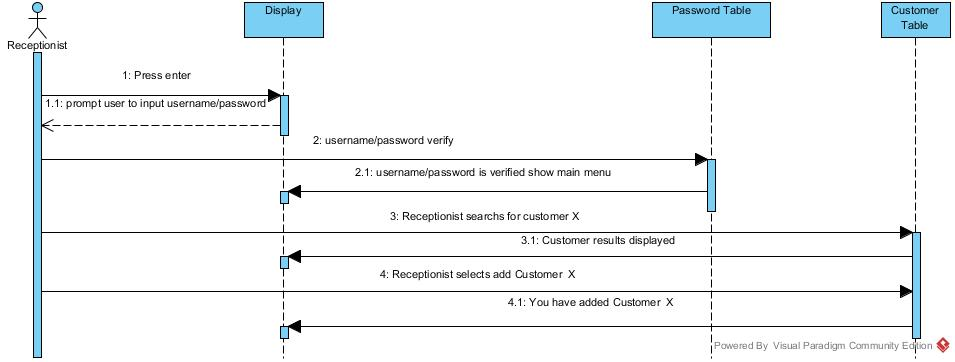
**Admin**

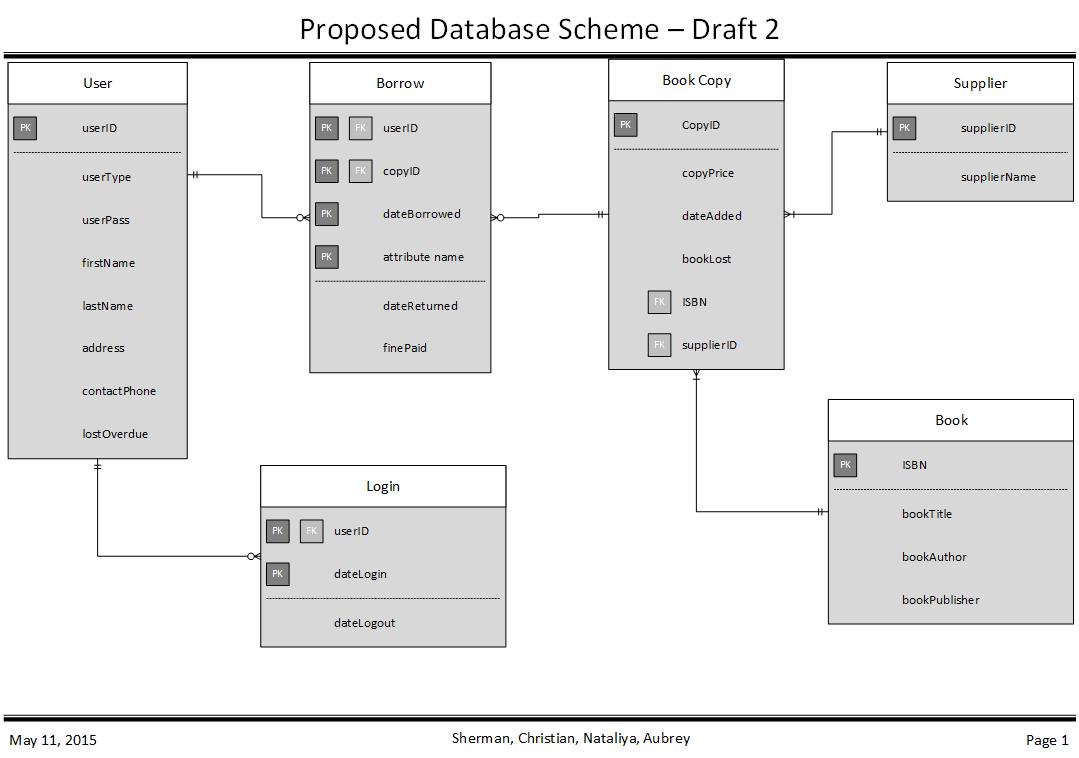
Example of admin searching and deleting book



**Receptionist**

Example of receptionist adding new customer





|  |  |  |
| --- | --- | --- |
| **USER** | |  |
| userId | PK int | 12345 |
| userType | Char | Admin |
| UserPassword | Variable character |  |
| firstName | Variable character |  |
| lastName | Variable character |  |
| Address | Variable character |  |
| contactPhone | int |  |
| lostOverdue | int |  |

|  |  |
| --- | --- |
| **Borrow** | |
| userID | PK, FK int |
| CopyID | PK, FK int |
| dateBorrowed | PK date |
| dateReturned | Date |
| finePaid | money |

|  |  |
| --- | --- |
| **Book Copy** | |
| CopyID | PK, int |
| Copyprice | Money |
| DateAdded | Date |
| BookLost | Boolean |
| ISBN | Int |
| SupplierID | FK, Variable character |

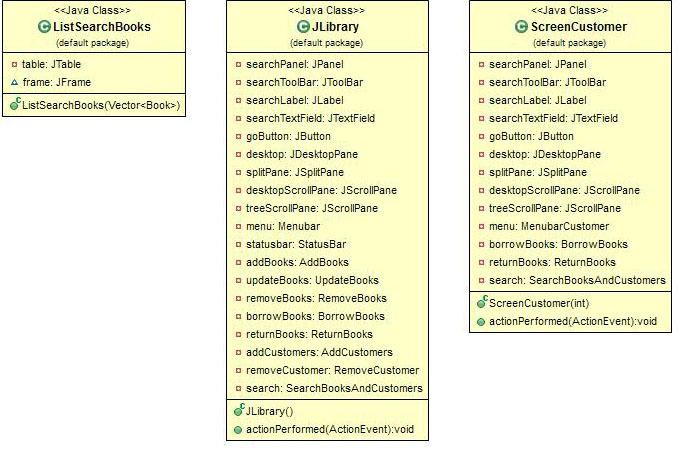
|  |  |
| --- | --- |
| **Supplier** | |
| SupplierID | PK, Variable character |
| SupplierName | Variable character |

|  |  |
| --- | --- |
| **Book** | |
| ISBN | PK, Int |
| bookTitle | Variable character |
| BookAuthor | Variable character |
| BookPublisher | Variable character |

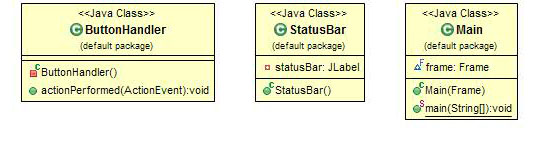
|  |  |
| --- | --- |
| **Login** | |
| userID | userID PK, FK |
| dateLogin | Date, PK |
| dateLogout | date |

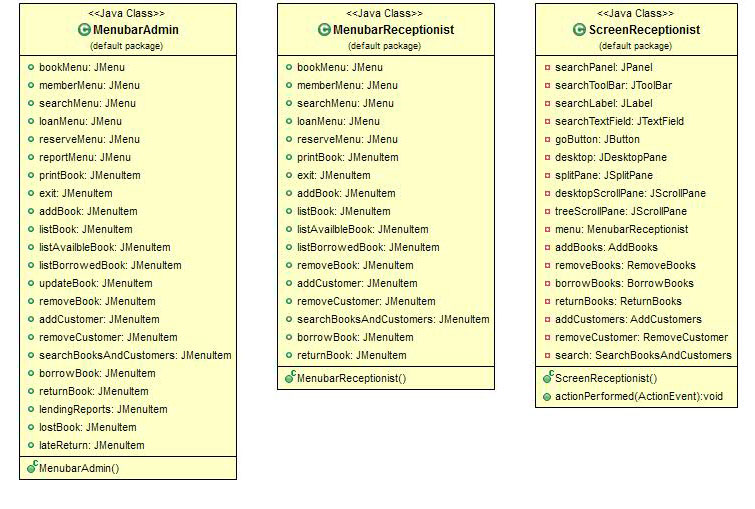
**Class Diagrams**

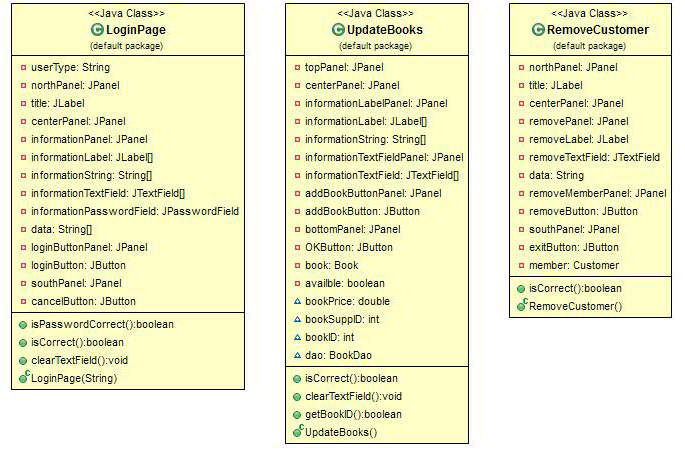
**GUI Classes**



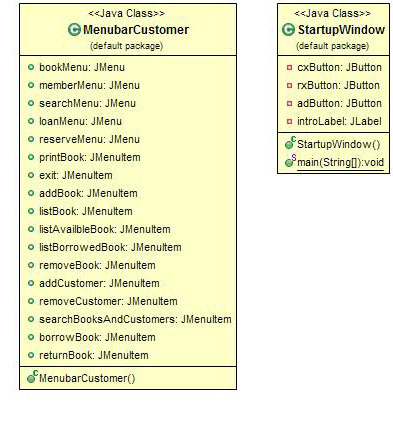




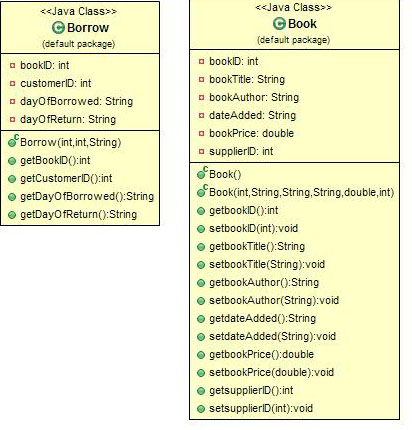


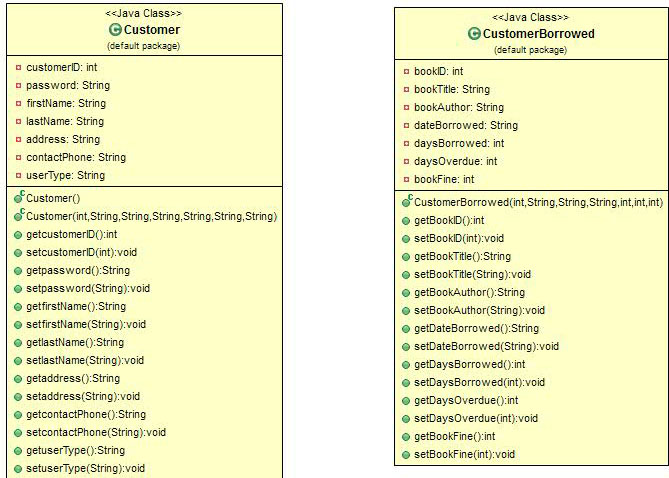




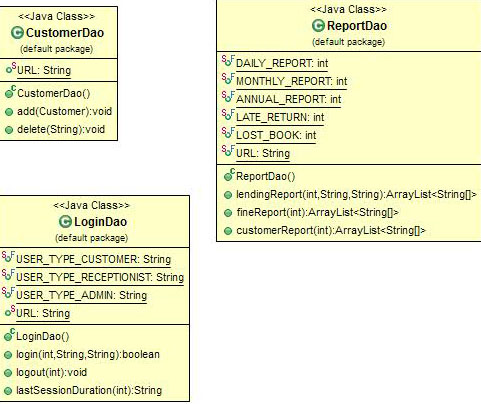


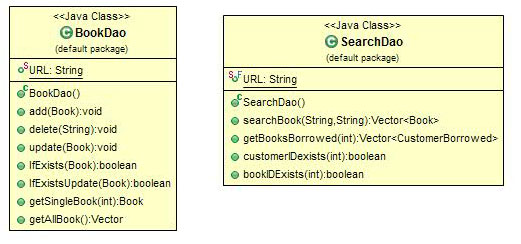
**Constructor Classes**

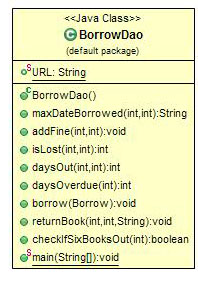




**DAO Classes**

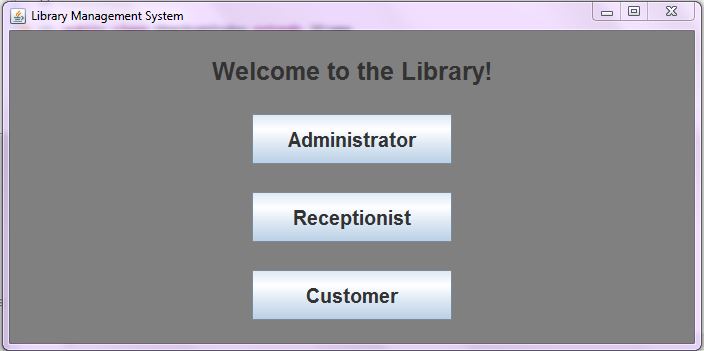
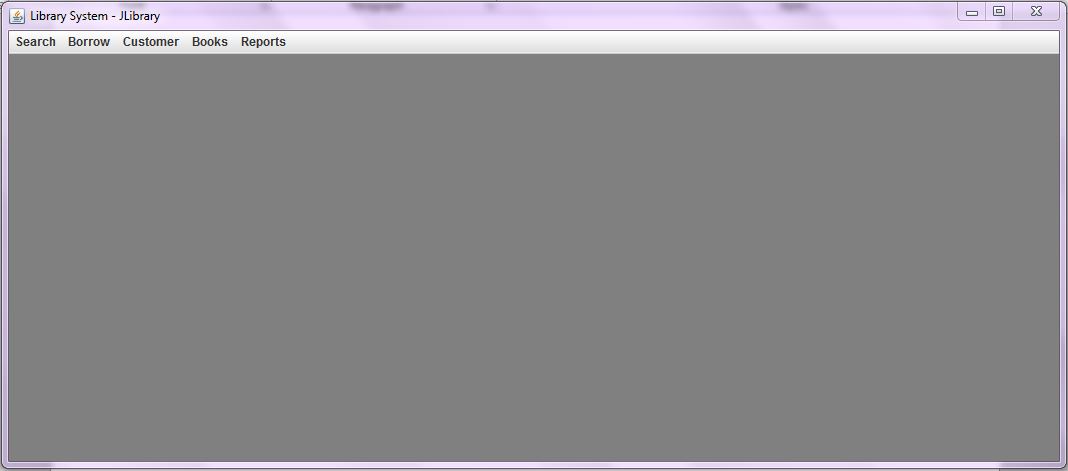




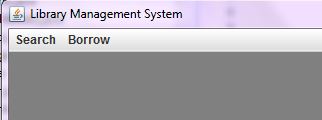


**Graphic User Interface Screens**

First Screen:

Main Administrator Screen:

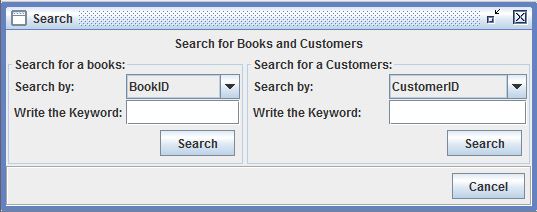
Customer Main Screen:

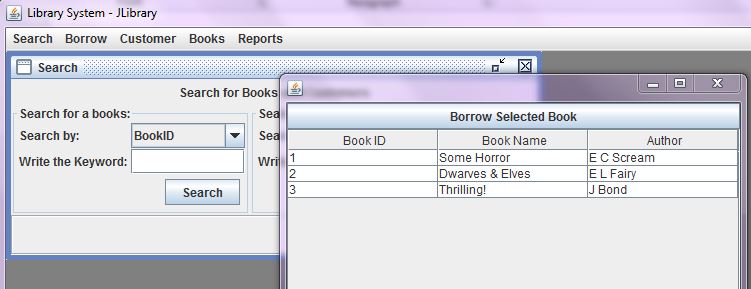


Login Screen:

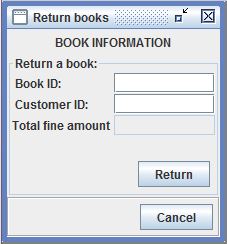
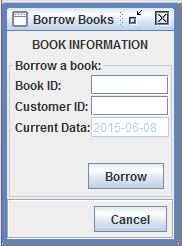


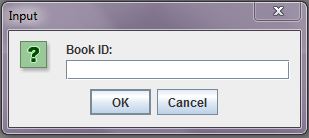
Search Screen:



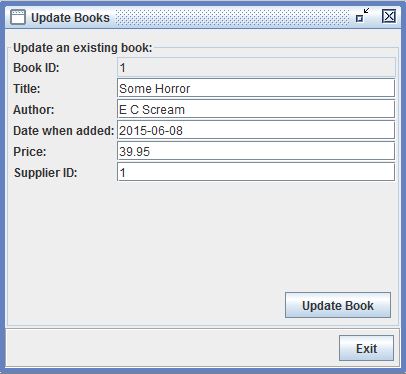
Displaying Searched Items Screen with Borrow Button:

Borrow Screen: Return Book Screen:

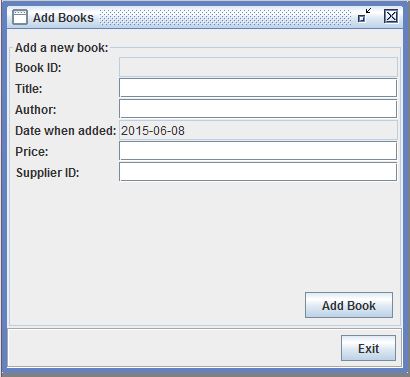




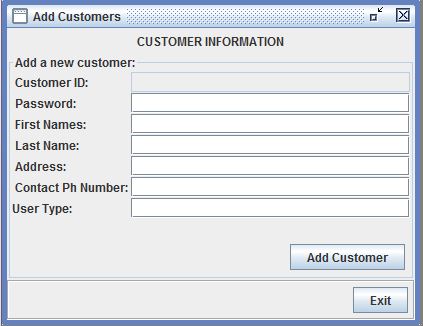
Dialog Screen for Input to go to Update Book:

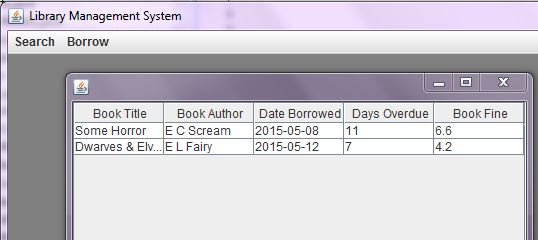


Add Book Screen:

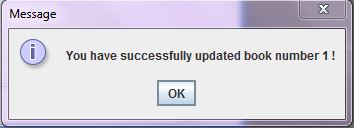
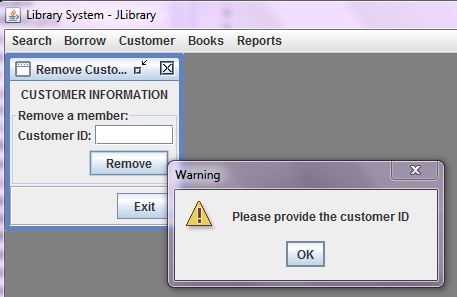
****

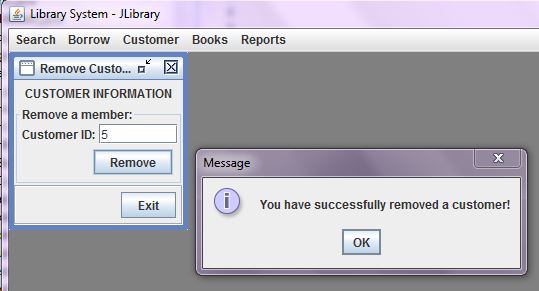
Add Customer Screen:



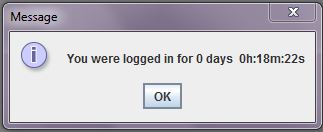


Customer Personal Account Screen:

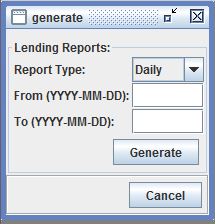
**** Prompt Screens:

****

Final Screen with Total Time:

****

Report Screen:

****

**Agendas and Minutes**

**Meeting 1**

Date 15/04/15 Time 2pm-3pm

Members presented:

Sherman

Nataliya

Christian

Aubry

Topics covered

* Discussed overall plan
* Broke assignment down into six parts :

1. GUI—Frontend. Where the user interacts with the system
2. Java Code-- Middlewhere that search is turned into a method
3. Database –-Backend where the method is turned into a queries.
4. Results of that query stored
5. Query results returned to a method
6. Method output**.**

Important decisions made

* To use SQL lite and to link the data base to Java rather than using binary files
* We need to research how these queries work with Java

**Meeting 2**

Date 22/04/15 Time 1pm-4pm

Members presented:

Sherman

Nataliya

Christian

Aubery

Topics covered

* Discussed what exactly we need (re wrote what is required as built points)
* Problem solving (reports and passwords) through SQL
* Flow charts
* Methods classes objects instance variables and how they will all interact

Problem solving topics discussed

* How do we separate users? i.e receptionist, admin, customer?
* How do we structure out SQL tables?
* How do we deal with lost books?
* How do we deal with overdue books?
* Do we have just a user class?
* What common functions will each user use? Ways to reduce code duplication.
* Do we have class for each user?
* What methods do we need and what will they look like?
* Do we need a password for each user?
* Each user will be a class which will have a username as an instance variable
* How can we each work on code separately – Git Hub? Google docs?
* What has been done before that we can base our design on what has been done previously?
* Can receptionists and admin also get books out? How would we do that?

**Meeting 3**

Date 11/05/2015 Time 12pm- 2pm

Members presented:

Nataliya

Christian

Sherman

Topics covered

* Who is doing what?
* Checkpoint report

**Meeting 4, Meeting 5, Meeting 6, Meeting 7, etc.**

Members presented:

* Sherman
* Nataliya
* Christian

**Contribution to the project:**

Sherman, Christian – Backend (Database, DAO)

Nataliya – Frontend (GUI), Documentation