Robot librarian

Group 4

Presentation Overview

Team distribution

Team Management and documentation

Abdullah Alhabib (371111359)

Modelling

Abdullah Alhasan (382124116)

Sultan Alodaib (391107956)

Sultan Alharbi (371111960)

Implementation

Mgren Almarshood (361111435)

Mothanna Almohaimed (391108661)

Group Management

Methods of communication

1. Whatsapp
2. Discord App

Repositories

1. Github
2. Discord App

Meeting pattern

When necessary

Software Process model

Incremental Development

- Suitable for our lack of experience
- Suitable for the nature of a semester
- Easy to fix mistakes

Where it falls short

- Harder to document the development
- Cross team communication is harder

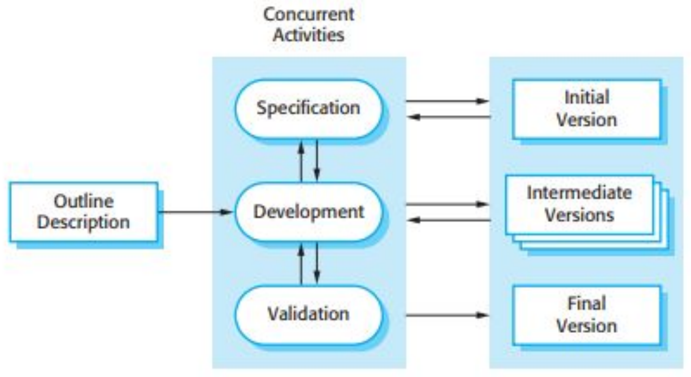
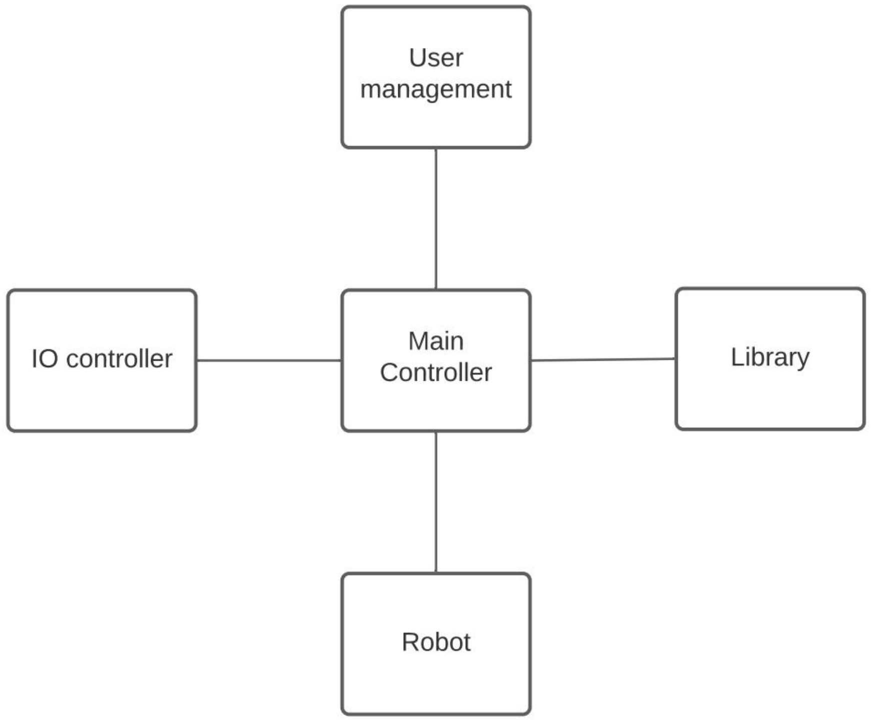


Figure 2.2 Incremental development

System Architecture

Goals from architecture

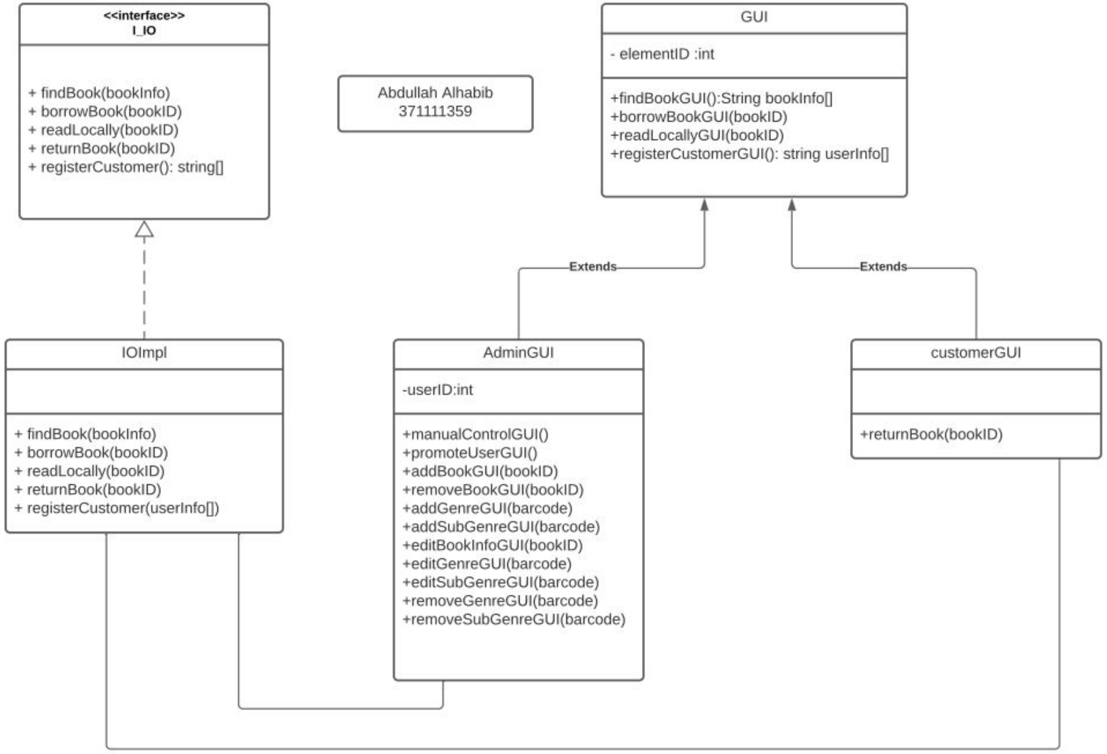
- Scalabile
- Modular
- Evolutionary



UML class Diagrams

Goals from design

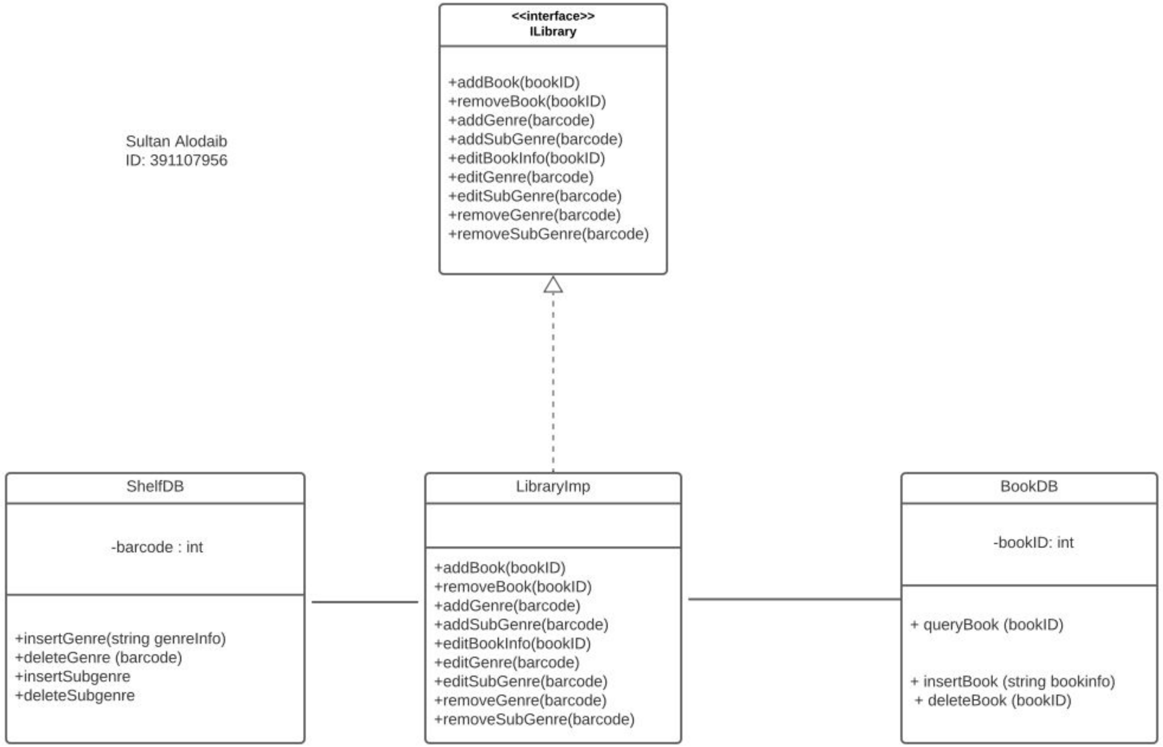
- Separate Admin from customer for security
- Create an interface between the GUI interface and the robot



UML class Diagrams cont

Goals from design

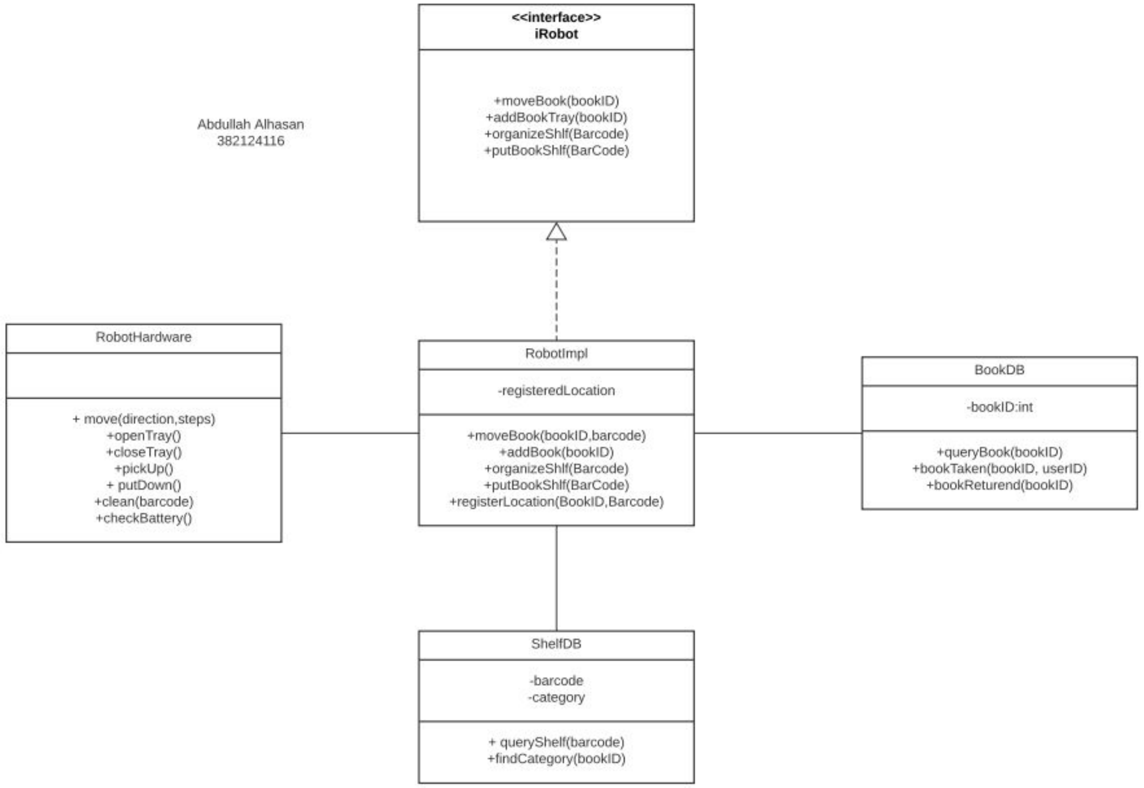
- Create an interface between the robot and library databases
- Allow for management of barcodes and books



UML class Diagrams cont

Goals from design

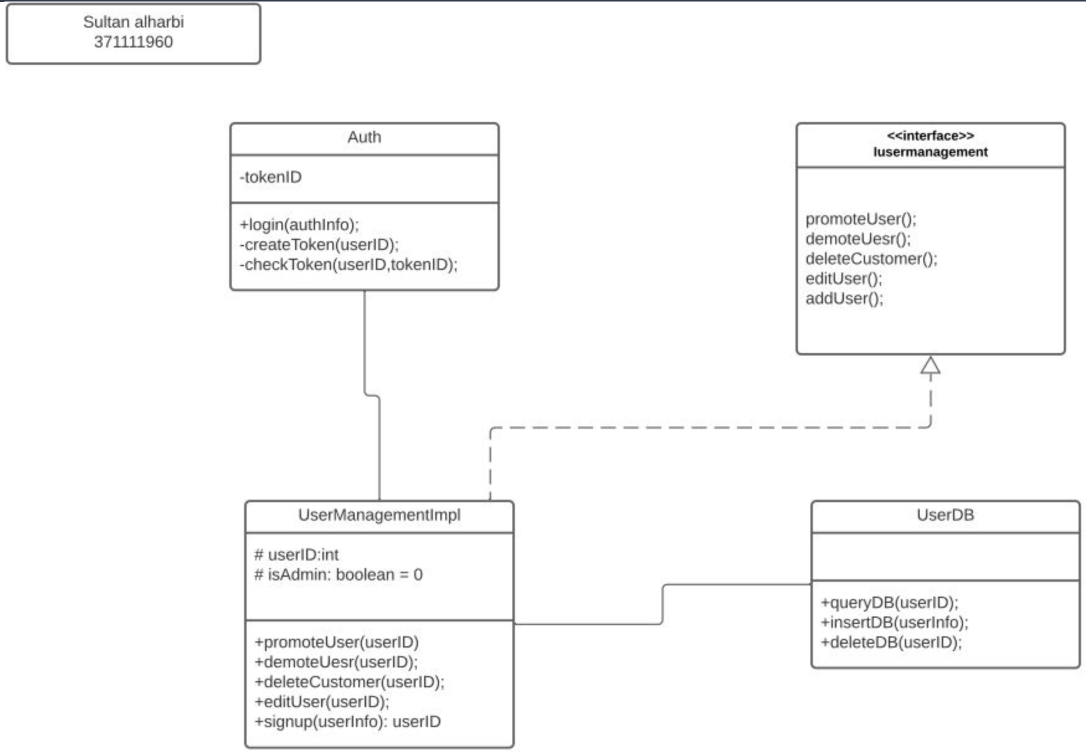
- Create an interface between the robot Hardware and our software
- Manages robot behaviour for cleaning, borrow, return and movement



UML class Diagrams cont

Goals from design

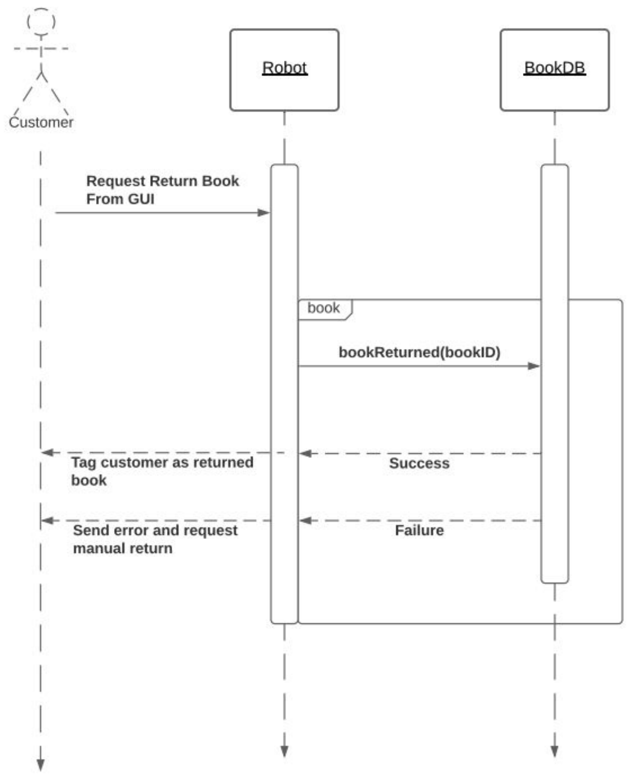
- Create an interface between the robot and Users
- Allows for some user management capability



Sequence Diagrams

Sequence for Return Book

1. Customer clicks on return book and puts book in tray
2. Robot sends book ID to bookReturned() in bookDB
3. For every book, BookDB makes sure everything is correct, then sends success or failure
4. For successful return, the customer is notified and tagged as returned in DB
5. For failure, the customer is asked to manually return the book.

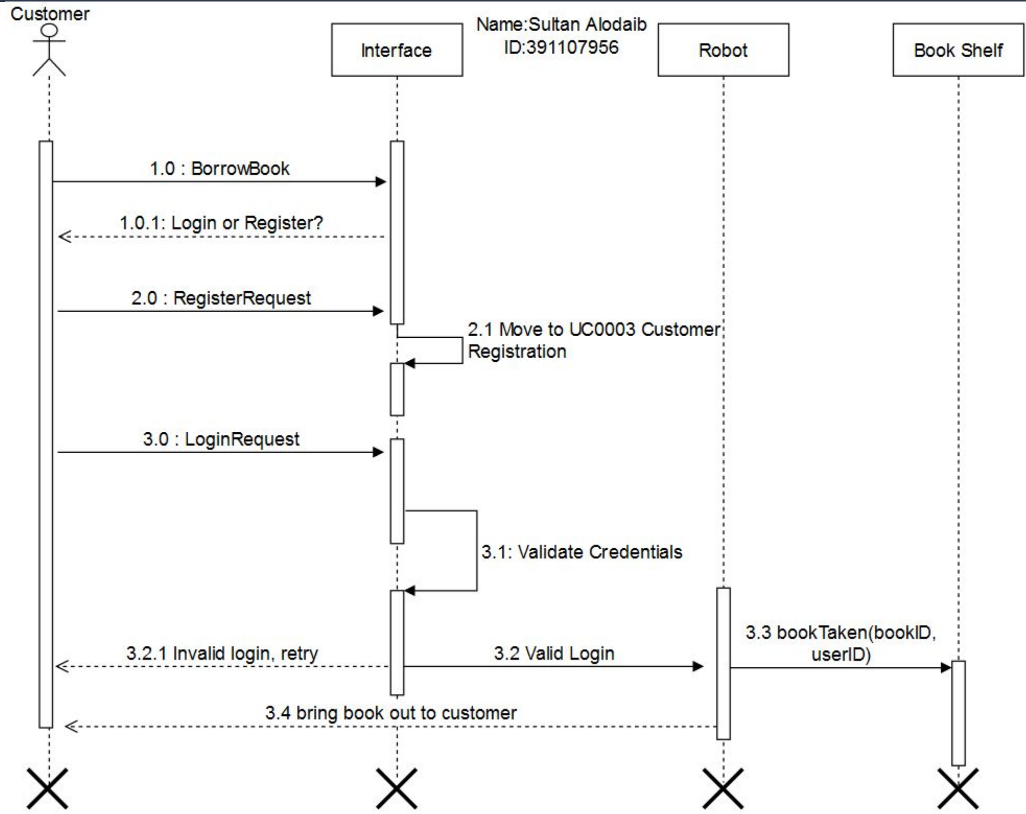

Customer
Robot
BookDB
Request Return Book From GUI
book
bookReturned(bookID)
Tag customer as returned book
Success
Send error and request manual return
Failure

Sultan ALharbi
371111960

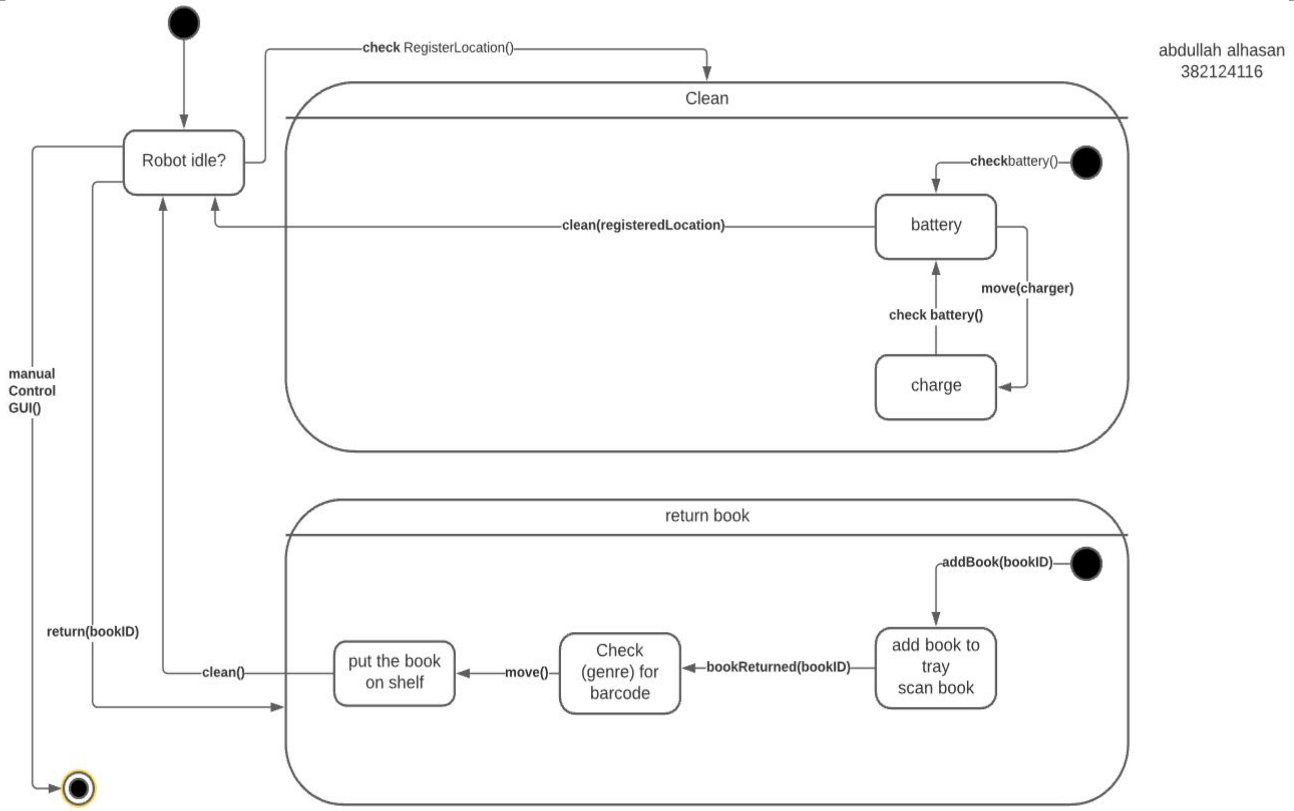
Sequence Diagrams cont

Sequence for Borrow Book

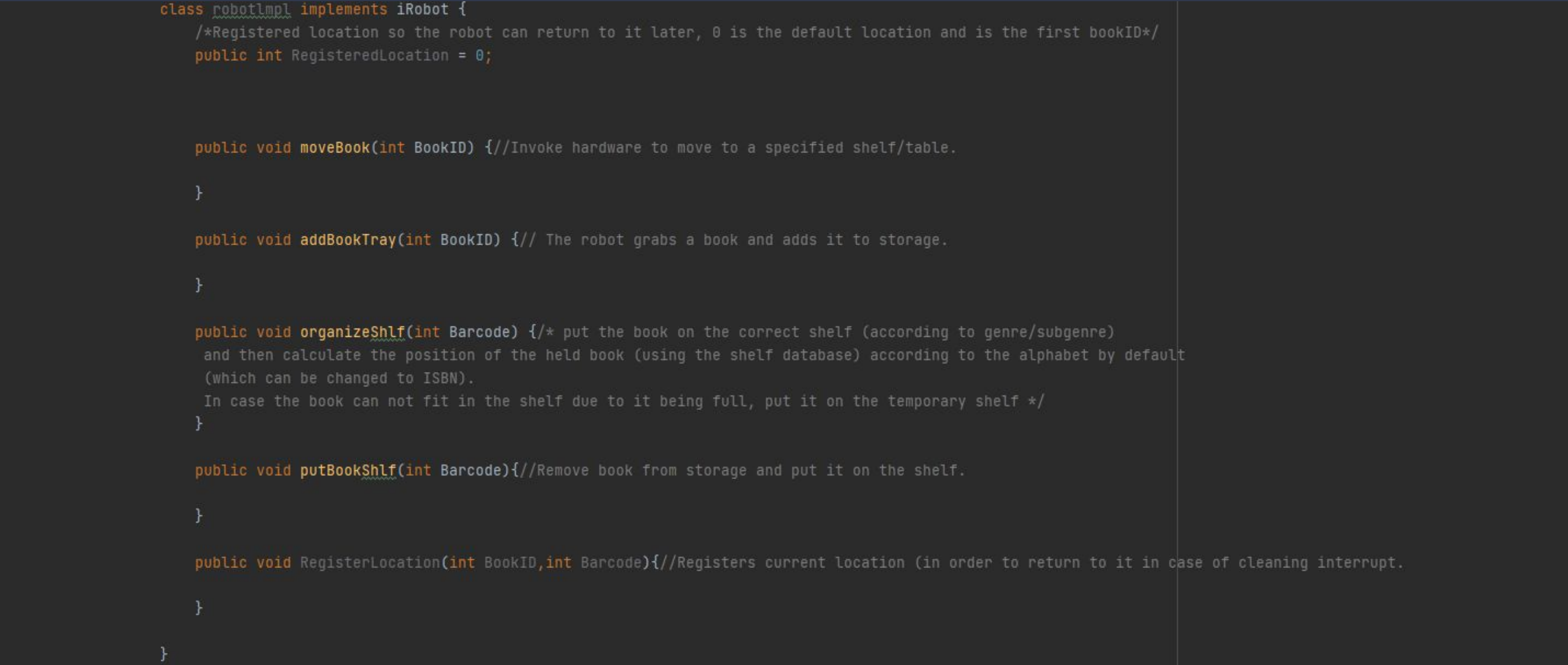
1. The customer clicks on the option to borrow the book from GUI
2. Customer is prompted to login or register
3. In case of register, move on to register customer process
4. In case of login, validate login then run bookTaken function that tags book and customer as borrowing
5. Move to book and take it back to customer.


Customer
Name:Sultan Alodaib
ID:391107956
Interface
Robot
Book Shelf
1.0 : BorrowBook
1.0.1: Login or Register?
2.0 : RegisterRequest
2.1 Move to UC0003 Customer Registration
3.0 : LoginRequest
3.1: Validate Credentials
3.2.1 Invalid login, retry
3.2 Valid Login
3.3 bookTaken(bookID, userID)
3.4 bring book out to customer

State Diagram

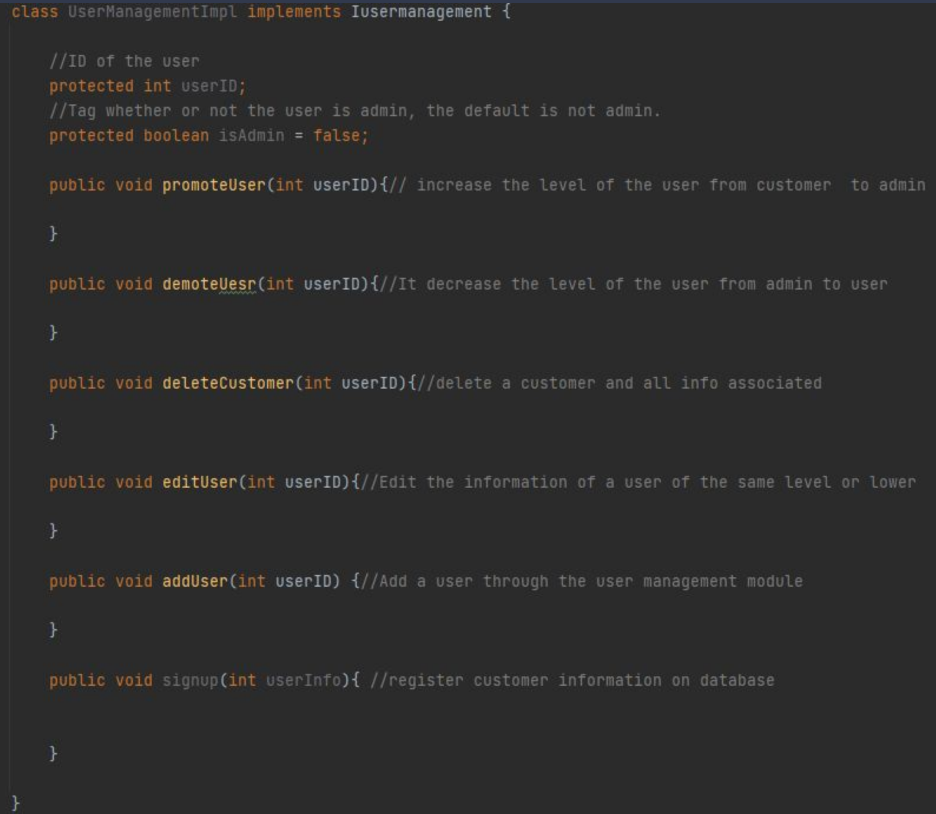


Implementation sample 1



```java
class robotImpl implements iRobot {
    /*Registered location so the robot can return to it later, 0 is the default location and is the first bookID*/
    public int RegisteredLocation = 0;



    public void moveBook(int BookID) {//Invoke hardware to move to a specified shelf/table.

    }

    public void addBookTray(int BookID) {// The robot grabs a book and adds it to storage.

    }

    public void organizeShlf(int Barcode) {/* put the book on the correct shelf (according to genre/subgenre)
     and then calculate the position of the held book (using the shelf database) according to the alphabet by default
     (which can be changed to ISBN).
     In case the book can not fit in the shelf due to it being full, put it on the temporary shelf */
    }

    public void putBookShlf(int Barcode){//Remove book from storage and put it on the shelf.

    }

    public void RegisterLocation(int BookID,int Barcode){//Registers current location (in order to return to it in case of cleaning interrupt.

    }

}
```

Implementation sample 2



```java
class UserManagementImpl implements Iusermanagement {

    //ID of the user
    protected int userID;
    //Tag whether or not the user is admin, the default is not admin.
    protected boolean isAdmin = false;

    public void promoteUser(int userID){// increase the level of the user from customer  to admin

    }

    public void demoteUesr(int userID){//It decrease the level of the user from admin to user

    }

    public void deleteCustomer(int userID){//delete a customer and all info associated

    }

    public void editUser(int userID){//Edit the information of a user of the same level or lower

    }

    public void addUser(int userID) {//Add a user through the user management module

    }

    public void signup(int userInfo){ //register customer information on database


    }
}
```

Thank you for listening!
Any questions?