CS383 Group Project

Semester #431

Software Design Document (SDD)

For Robot Librarian

Version 1.1

Prepared by Group 4

Change history	
version	**Changes**
1.0	Initial document release
1.1	Added contents section

Content

1 Introduction

This must contain an overview of the SDD and a description of the scope of the system to be developed.

1.1 Scope

The project is a requirement for the software engineering course at the college of computer in AlQassim university. It aims to create a software interface between a robot and a library, the specifics of the robot and the library are out of the scope, instead we're focusing on the interface between them, and the application of a robot librarian.

1.2 Document Structure

This SDD explains the system following these steps:

- Show a diagram
- Brief overview explanation for the diagram
- Detailed explanation

1.3 constraints

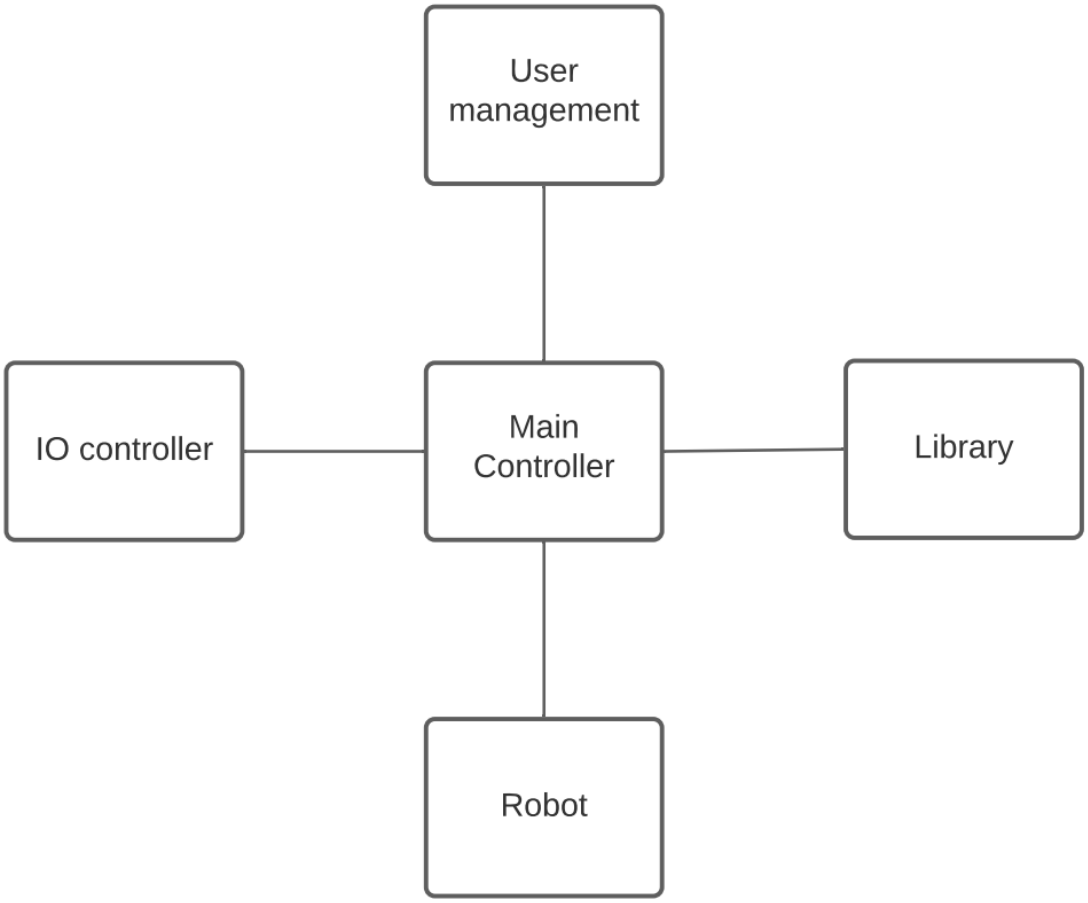
- Focus constraint: Students are not solely doing this project therefore some loss of focus can cause things to lose prospect
- It is hard to create an interface without being very familiar with the external parts of the interface (Robots and libraries) and since we are students it is hard to familiarize ourselves with them for this particular project.
- Lack of experience, while it is expected, it did cause major drawbacks in our design, but we have learned a VERY great deal from them and we have made mistakes we are sure to avoid in the future.

2 System Overview

The system is an interface that allows a robot to do some librarian tasks, it was design following the incremental development software activity process model, since it is suitable for our lack of experience as students, suitable for the nature of a semester and makes it easy to fix mistakes, since we can go back and forth between development teams

3 System Architecture and Components Design

3.1 Architectural design



The system consists of:

- Main controller that connects to everything with the main function
- Library module that manages book and shelf databases
- Robot module that manages the interface between the system and the robot hardware
- User management module for some user management capabilities and security
- IO controller for managing information input and output with the Graphical User Interface (GUI)

3.2 Component Decomposition Description

1 Main controller

Is simply the main function that ties everything together.

2 Library module

This module divides into:

- **Main library**
- **ShelfDB controller**
- **BookDB controller**

3 Robot Controller

- **Main robot controller**
- **ShelfDB**
- **BookDB**
- **Robot Hardware**

4 User management module

- **main user management controller**
- **UserDB**
- **Auth controller**

5 IO controller

- **Main IO controller**
- **Admin GUI controller**
- **Customer GUI controller**

3.3 Detailed Components Design Description

Component Identifier: C1 Main controller

Purpose:
Run the system.

Function:
Manage Robot state and run the correct components, and error handling.

Subordinates:
None.

Dependencies:
All other components

Component Identifier: C2 Library Module

Purpose:
Section 3.3.3 of SRS

Function:
Implement the interface between the robot and library databases

Subordinates:
Main controller, Robot controller

Dependencies:
Shelf Database and Book database.

Data:
Input and output of shelf database and book database

Component Identifier: C3 Robot controller

Purpose:
Section 3.3.2 of SRS

Function:
Utilize the robot hardware interface to make the robot complete commands

Subordinates:
IO controller

Dependencies:
Library module, Robot hardware interface, User management module

Interfaces:
Robot Hardware

Data:
Shelf barcodes and book IDs for borrow and return.

Component Identifier: C4 User management module

Purpose:
Section 3.3.4 of SRS

Function:
Provide some user management capabilities

Subordinates:
Main controller, Robot Controller, IO controller

Dependencies:
User Database

Data:
User data and authentication data

Component Identifier: C5 IO controller

Purpose:
Section 3.3.1 of SRS

Function:
Interface with the external GUI components

Subordinates:
All components

Dependencies:
All components

Interfaces:
External GUI, remote GUI

Data:
Book data input, Customer data Input, status data Output

3.4 Design Rationale

The system was designed this way to accommodate for:

- **Usability**: extensive GUI design to accommodate useability
- **Manageability**: Some database management make for easily managed system
- **Scalability**: The Robot controller can support multiple robots, creating the option to expand robots.
- **Security**: Separation between the admin GUI and the customer GUI in IO controller allow for security.

4 Data Design

While this system does not have extensive database management, it still relies on external databases to function, the external databases should have the following in consideration:

4.1 Database Desctiption

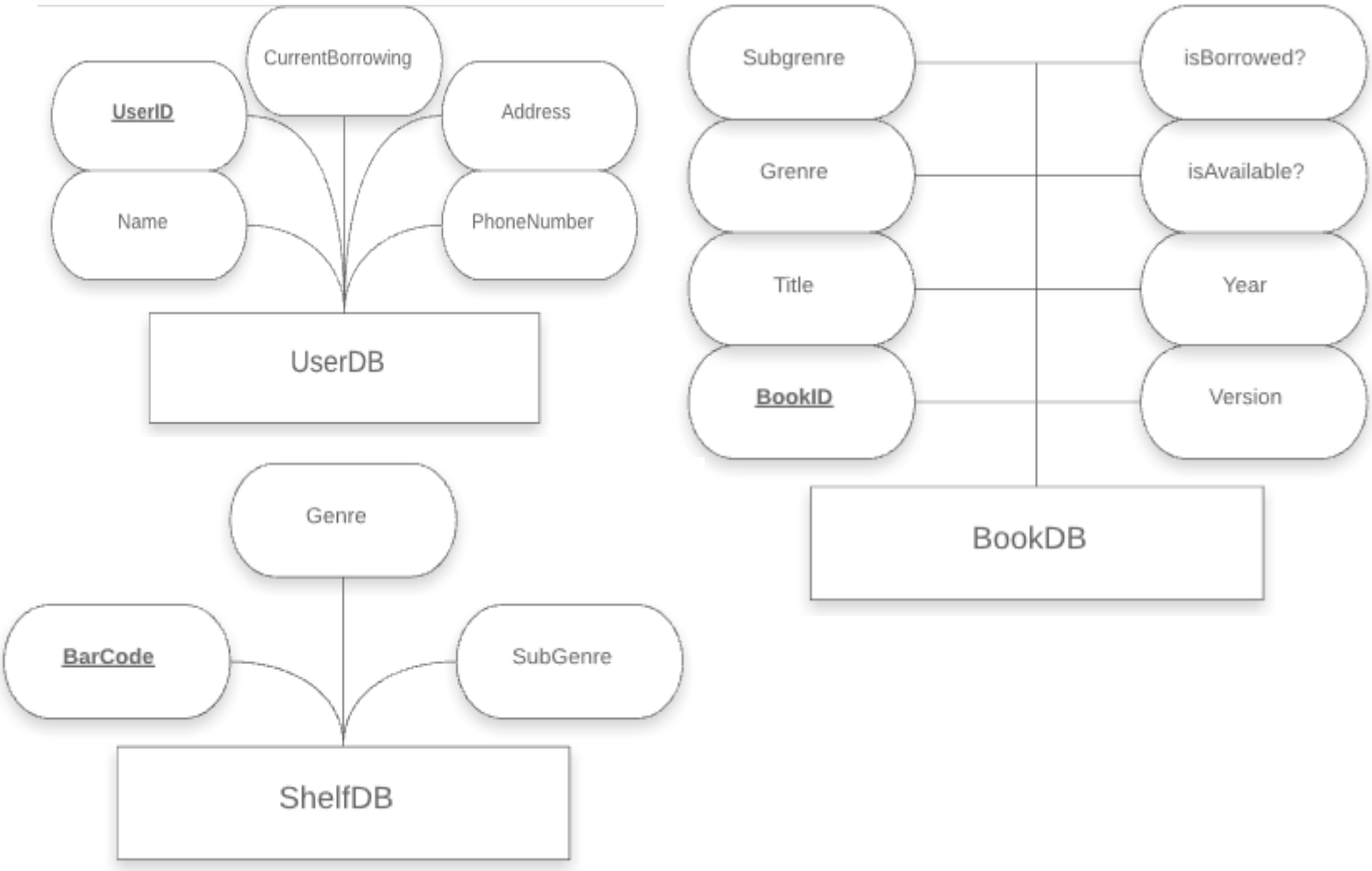
- **Book Database:**

There must be a database for books, using barcodes generated using ISBN as a unique ID, and containing title, genre, version and production year.

- **Shelf Database:**

The library must physically have barcodes assigned on shelves, according to a shelf database, where the barcode is the unique ID and each shelf belongs to a subgenre, which belongs to a genre.

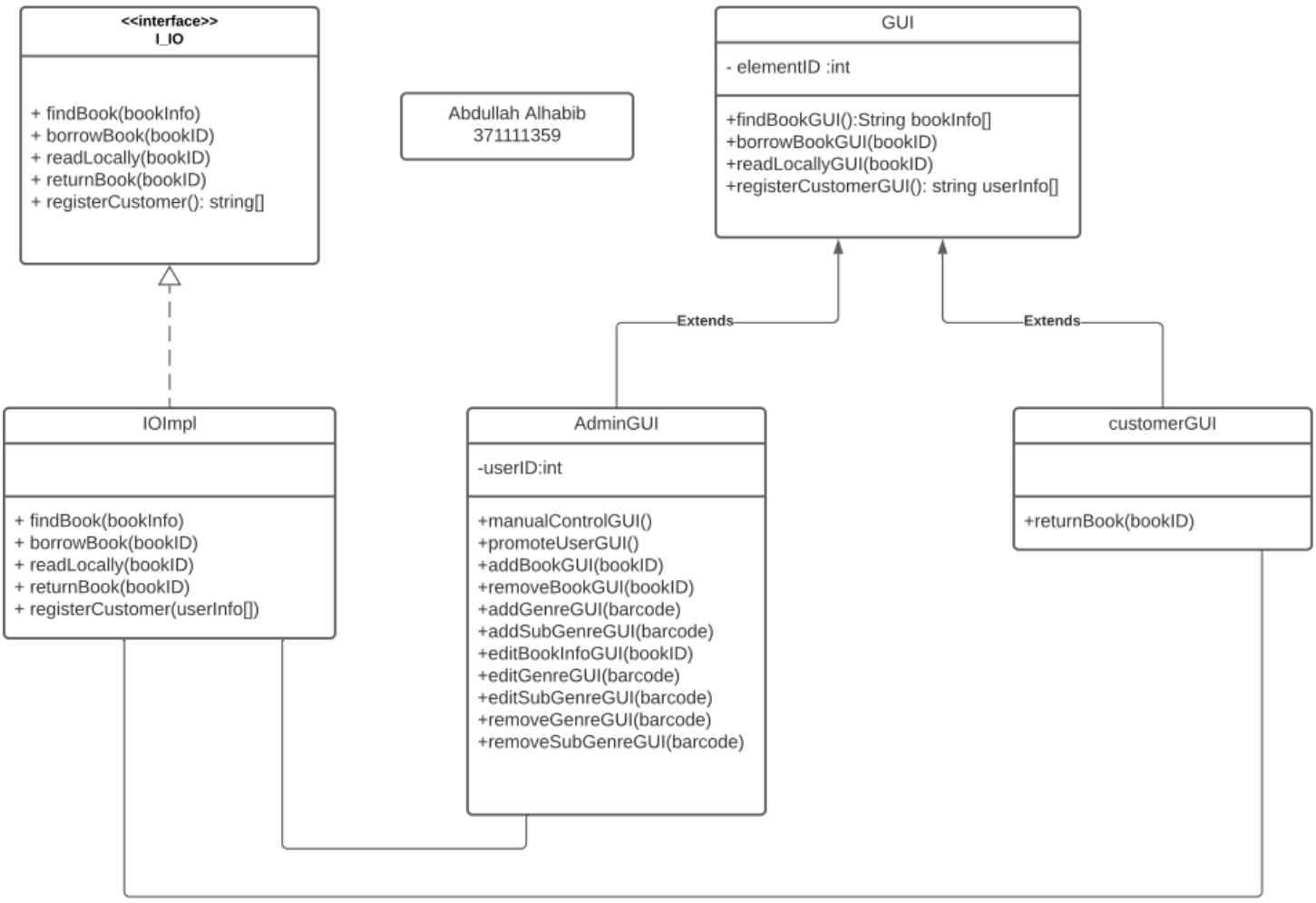
4.2 Data structures



5 Design Details

5.1 Class Diagrams

5.1.1 IO class diagram



Rationale:

While the detailed description of the functions are discussed in the SRS, the rationale for this design is the following:

Create a parent class GUI, so we can separate Admin and Customer GUIs for increased security, and the functions in these classes merely apply functions of the rest of the system to the GUI.

Details:

Details of operations and attributes are as follows:

IOImpl:

- **findBook(bookInfo):** This operation finds a book through searching the database using the book info provided.

- **borrowBook(bookID):** run the borrow book operation using the associated bookID.
- **readLocally(bookID):** request the book for local reading, the robot returns to the table that started the request.
- **returnBook(bookID):**Initiate the return book sequence discussed further in section.
- **registerCustomer(userInfo[]):** register the customer using the user info returned from the GUI.
- **returnBook():** initiate the return book sequence (SRS UC0005).

GUI:

All of the GUI functions are meant to display options on the GUI and take input, or display outputs on the GUI.

- **elementID:** The ID of the graphical element.
- **findBookGUI():** Display the option “find book” and display a form after clicking it, the form is returned to the findBook(bookInfo) function as input.
- **borrowBookGUI(bookID):** The system initiates the borrow sequence (SRS UC0002).
- **readLocallyGUI(bookID):** Display the option to “Read Locally” in the GUI, and after clicking it the program initiates the readLocally sequence (SRS UC0004).
- **registerCustomerGUI():** launches a form for the customer to fill in customer information and then return that information to registerCustomer(userinfo) above.

AdminGUI:

Inherits the GUI superclass to separate from the customer GUI.

- **manualControlGUI()**:Initiate manual control
- **promoteUserGUI():** Initiate promoteUser(userID) in the user management module
- **addBookGUI():** bring out a form to fill book info
- **removeBookGUI(bookID):** Tag a book as removed from the library.
- **addGenreGUI():** bring out a form to fill genre info.
- **addSubGenreGUI()**: bring out a form to fill subgenre info.
- **editBookInfoGUI(bookID):** edit a specific book’s info.
- **editGenreGUI(barcode):** edit a genre.
- **editSubGenreGUI(barcode):** edit a subgenre.
- **removeGenreGUI(barcode)**: Tag a genre as removed.
- **removeSubGenreGUI(barcode)**: Tag a subgenre as removed.

Notice that book, genre and subgenre removal only tags them for removal, limiting the potential damage from misuse, and limiting that functionality to original database control.

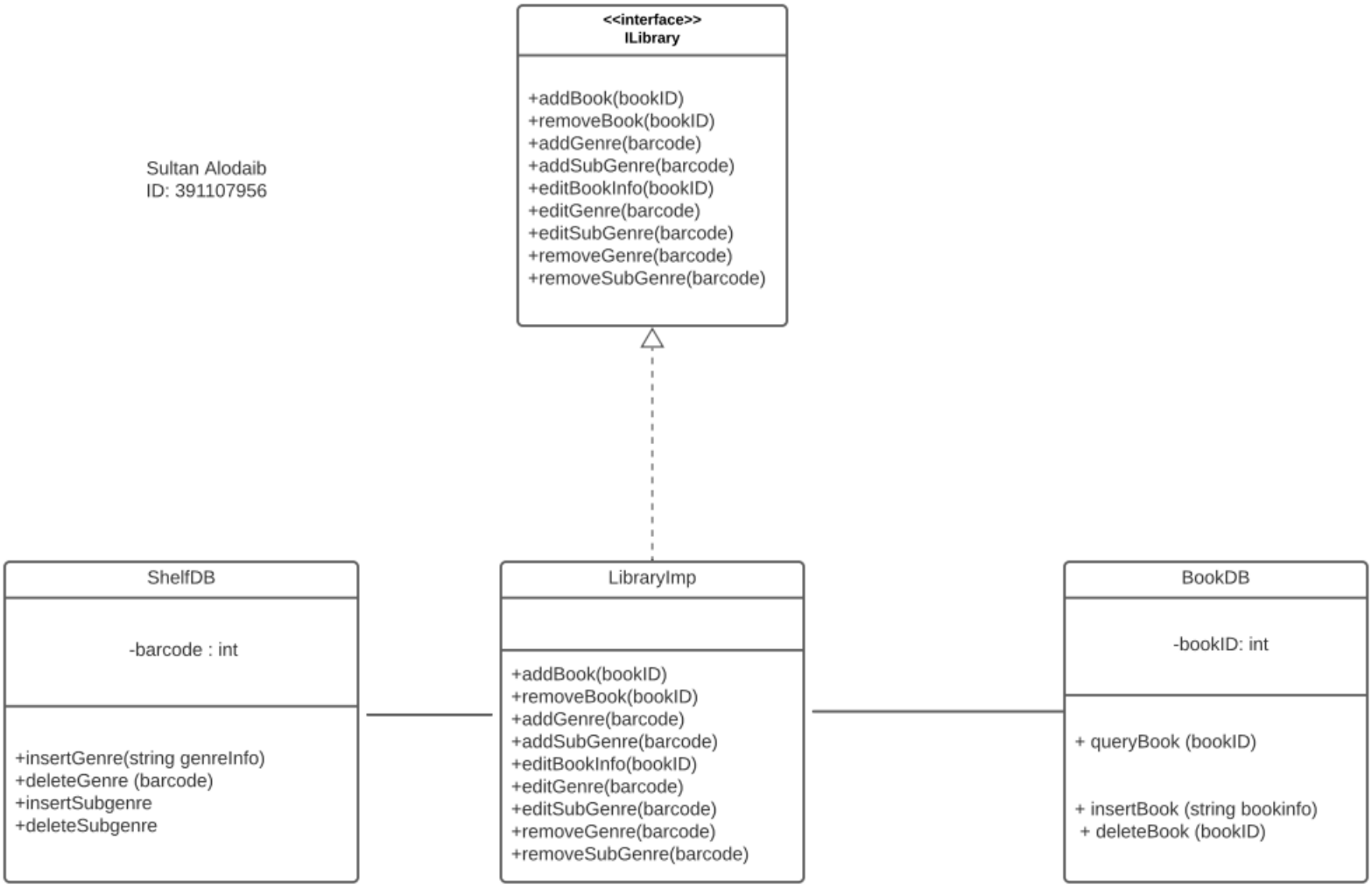
CustomerGUI:

Displays the functions of GUI only, without access to the admin GUI.

returnBook(bookID): initiate the return book sequence (SRS UC0005)

5.1.2 Library Class diagram

Sultan Alodaib
ID: 391107956



Rationale:
The library module controls the data of the shelf and book databases, it is only accessible by admins so the functions here are initiated by admin input from AdminGUI

Details:
Details of operations and attributes are as follows:

LibraryImpl:

- **addBook(bookID):** adds a book to the library.
- **removeBook(bookID):** Tag a book as removed from the library.
- **addGenre(genreInfo):** adds a genre to the shelf database.
- **addSubGenre(subgenreInfo)**: adda subgenre to the shelf database.
- **editBookInfo(bookID):** edit a specific book's info.
- **editGenre(barcode):** edit a genre.
- **editSubGenre(barcode):** edit a subgenre.
- **removeGenre(barcode)**: Tag a genre as removed.
- **removeSubGenre(barcode)**: Tag a subgenre as removed.

Notice that book, genre and subgenre removal only tags them for removal, limiting the potential damage from misuse, and limiting that functionality to original database control.

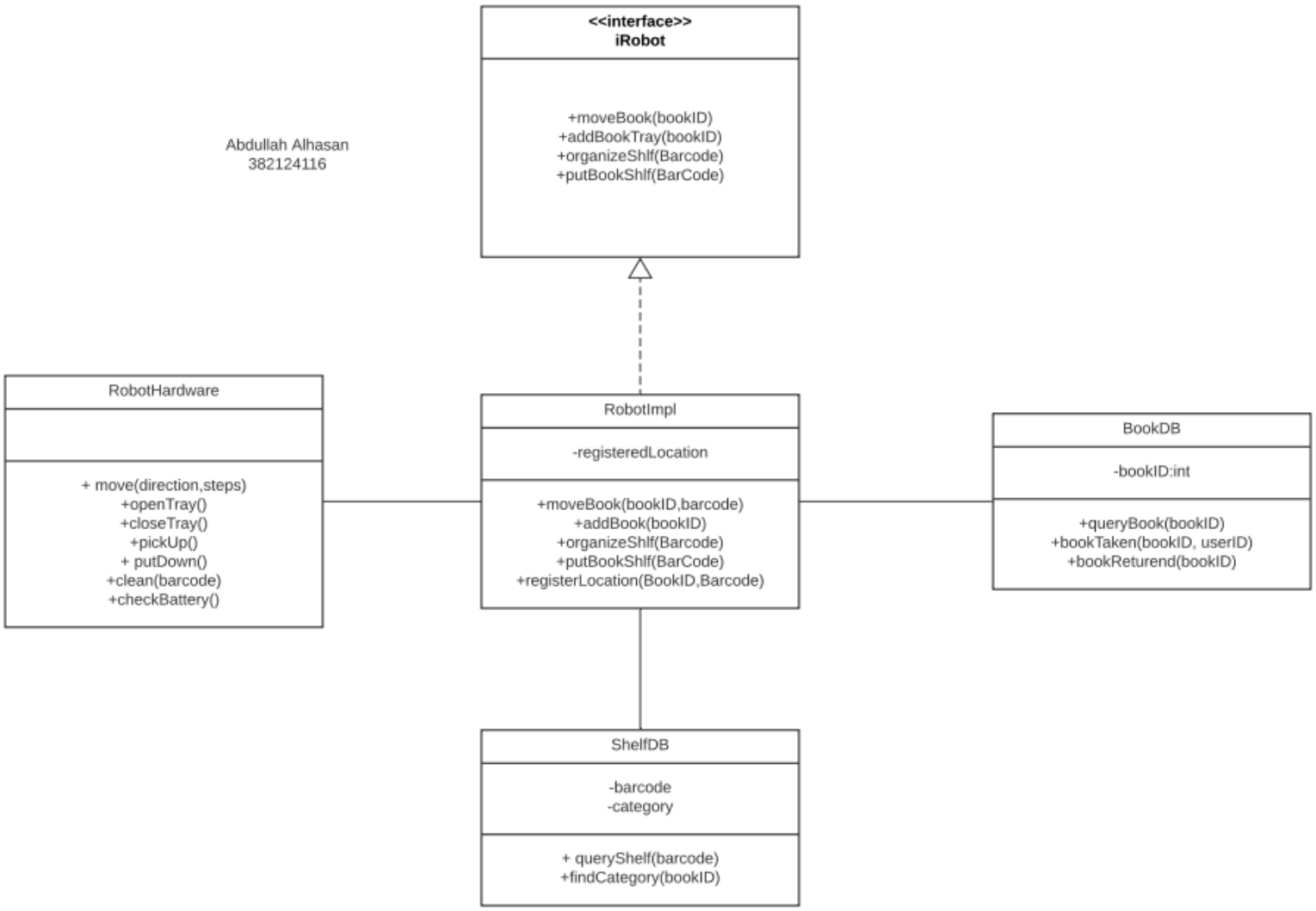
ShelfDB:

- **insertGenre(string genreInfo):** Create a new genre in the database.
- **deleteGenre (barcode):** tags a genre removed in the database.
- **insertSubgenre(subgenreInfo)**:Create a new subgenre in the database
- **deleteSubgenre(barcode)**: tag a subgenre as removed in the database.

BookDB:

- **queryBook (bookID):** retrieve book info
- **insertBook (string bookinfo):** insert a new book with info
- **deleteBook (bookID):**tag a book as removed.

5.1.3 Robot class diagram



Rationale:
Create an interface between the system and the external robot hardware interface, this controller focuses on moving the robot and changing between its modes.

Details:
Most of the operation happening here is automatic and shouldn't need any manual input, while the bot is not doing any operation it is considered idle and will go to cleaning mode, starting from the **registered location**, which is the ID of the book to be cleaned next, it is 0 by default, which means clean from the start.

RobotImpl:

- **registeredLocation**: the ID of the book to be cleaned next, it is 0 by default, which means clean from the start.
- **moveBook(bookID,barcode)**: move book to a shelf.
- **addBook(bookID)**: Add the book to local storage.

- **organizeShlf(Barcode)**:Organize a shelf according to the organize shelf use case discussed in SRS UC0006
- **putBookShlf(BarCode)**: put the book in the desired shelf.
- **registerLocation(BookID,Barcode):** register the location of the book to be cleaned next and store it on registeredLocation.

RobotHardware:
Interacts with the robot hardware provided externally.

- **move(direction,steps)**: invoke the robot hardware to move in a direction and steps, or a preset location like the barcode of a shelf
- **openTray() :**Open the book tray in the robot
- **closeTray()** :Close the book tray in the robot.
- **pickUp():** pick a book up
- **putDown():** put the book down
- **clean(barcode):**Clean a book
- **checkBattery()**: Check the battery level and act accordingly.

ShelfDB:
Database management

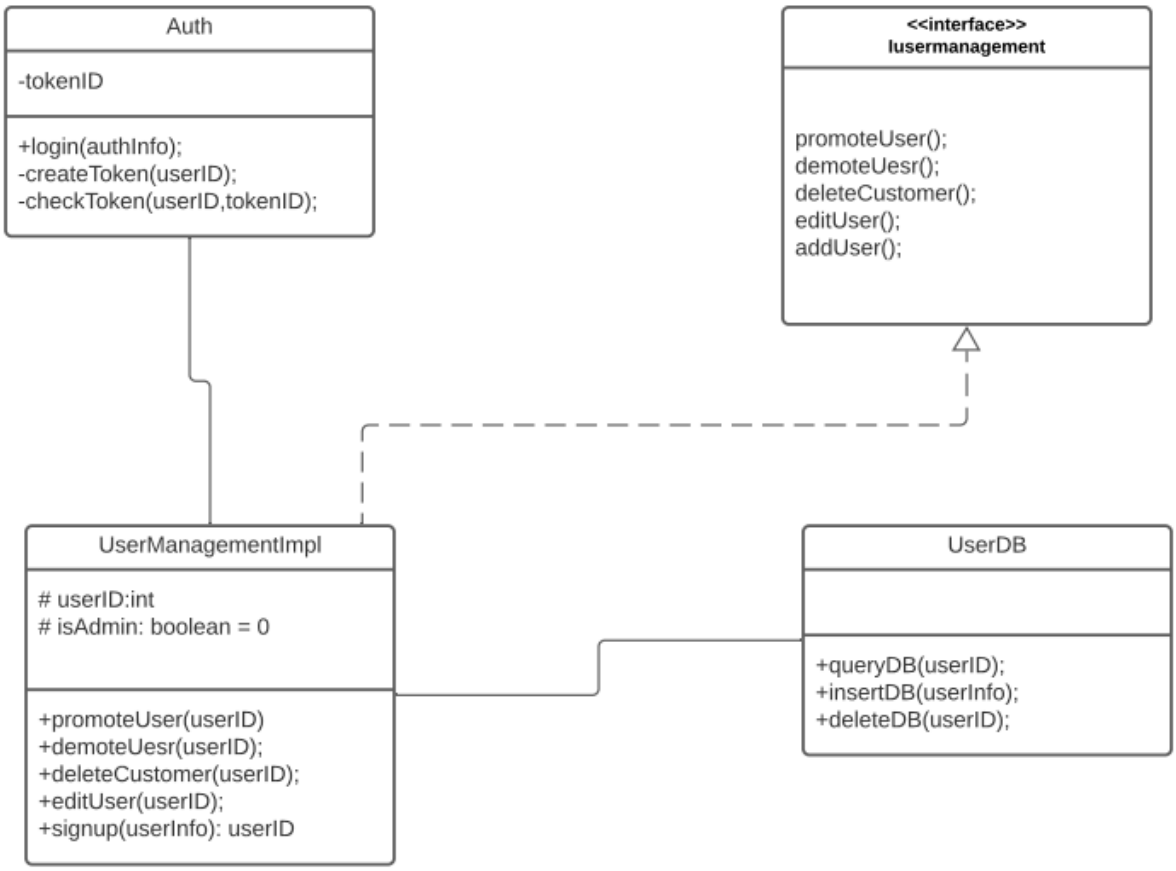
- **barcode :**barcode value.
- **Category:** category value.
- **queryShelf(barcode):** retrieve shelf data.
- **findCategory(bookID)**: retrieve category barcode and location.

BookDB:

- **bookID:** ID value of the book.
- **queryBook(bookID)**:Retrieve book info from the database
- **bookTaken(bookID, userID):** tag the book as taken and the user as the user that borrowed this specific book.
- **bookReturend(bookID):** tag the book as returned and the user associated as having returned the book.

5.1.4 User management class diagram

Sultan alharbi
371111960



Rationale:
Basic user management capabilities, since user management isn't within the scope of the system then what's here is the basics needed for the system to work, but it can be expanded upon for a more complete user management system.

Details:

Auth :

- **tokenID:** hashed auth token value**.**
- **login(authInfo):** Validate the login process
- **createToken(userID):** Create the auth token for the user
- **checkToken(userID,tokenID):**Check the validity of the auth token

userManagementImpl:

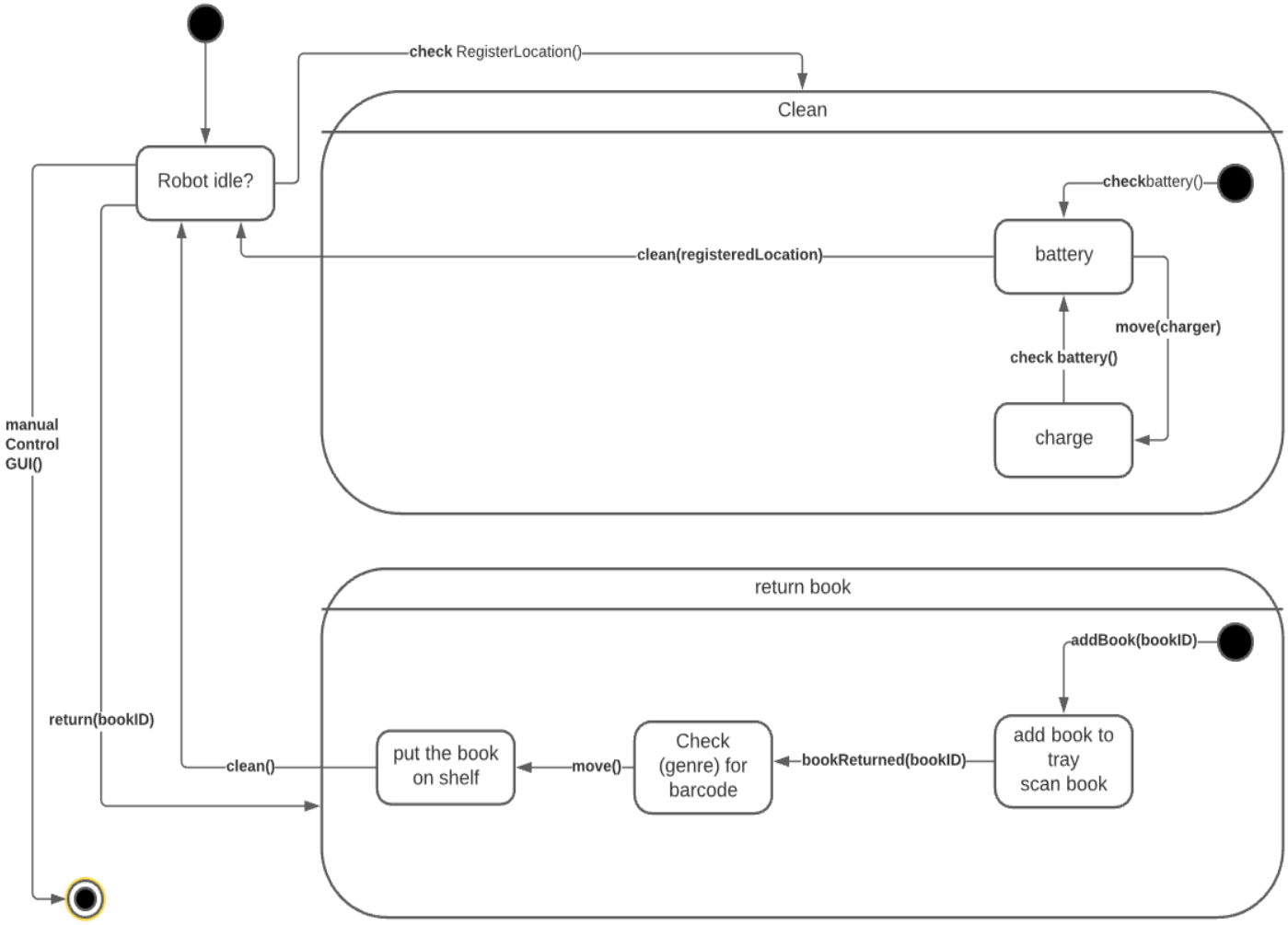
- **userID:** ID of the user
- **isAdmin:** tag whether or not the user is an admin, the default is false.
- **promoteUser(userID):**Promote user to admin status
- **demoteUesr(userID):**Demote user from admin status
- **deleteCustomer(userID):** tag a user as removed (notice it is not direct deletion to prevent database damage from misuse.
- **editUser(userID):** edit user info.
- **signup(userInfo):** Create a new user.

UserDB:

- **queryDB(userID):** retrieve the user info.
- **insertDB(userInfo):** insert a user to the database.
- **deleteDB(userID):** Tag a user as removed.

5.2 State diagram:

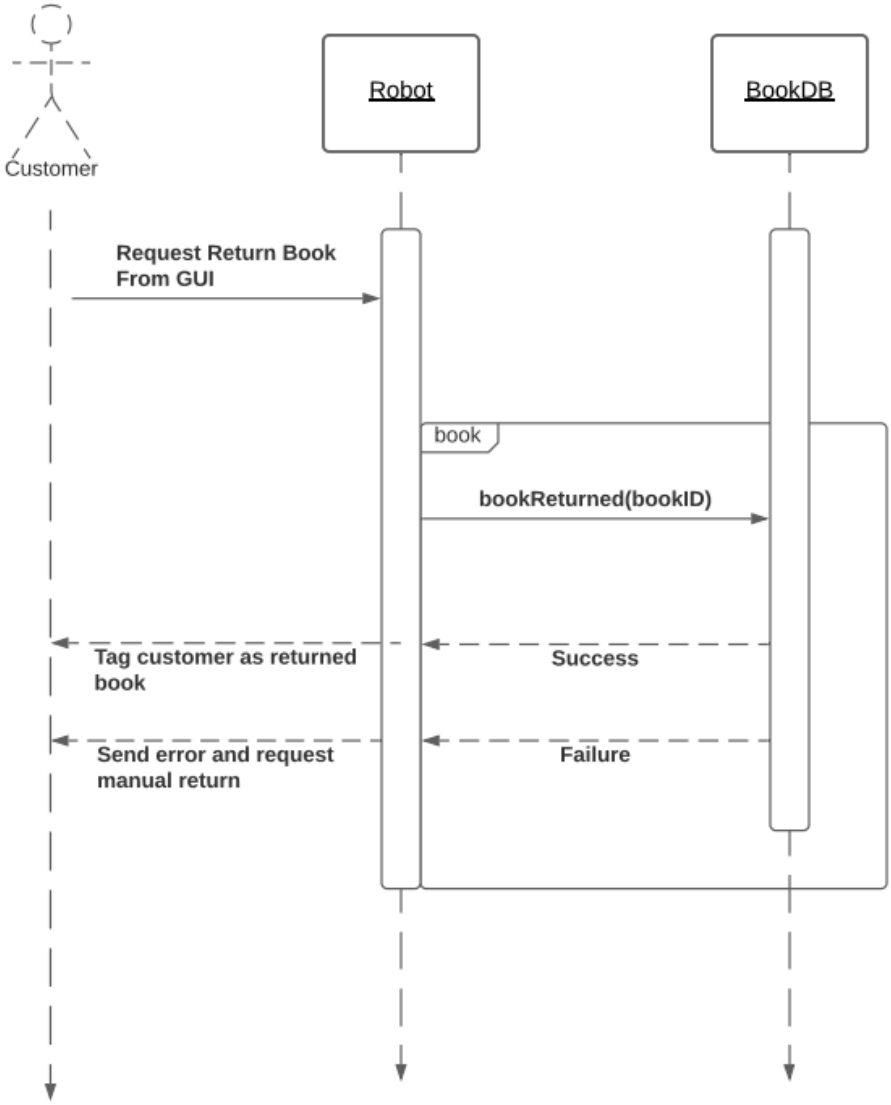
Clean-return state diagram:



- If the robot is idle, initiate clean process:
    1. Check battery
    2. if it's less than the time limit (default 1 hour) then move() to charger.
    3. Otherwise Clean() the registeredLocation, which is the location of the book to be cleaned next, which after finishing cleaning will call register location()
    4. Then check if the robot is idle and repeat
- If there is a request to return book, the robot will do it immediately:
    1. Addbook() puts the book in the robot's tray and scans the book.
    2. Then call bookReturned to tag the book returned and the customer has returned the book.
    3. Then check the genre for barcode by using queryShelf(bookID)
    4. Move() to that location and call putbookshlf() to put the book in the shelf.

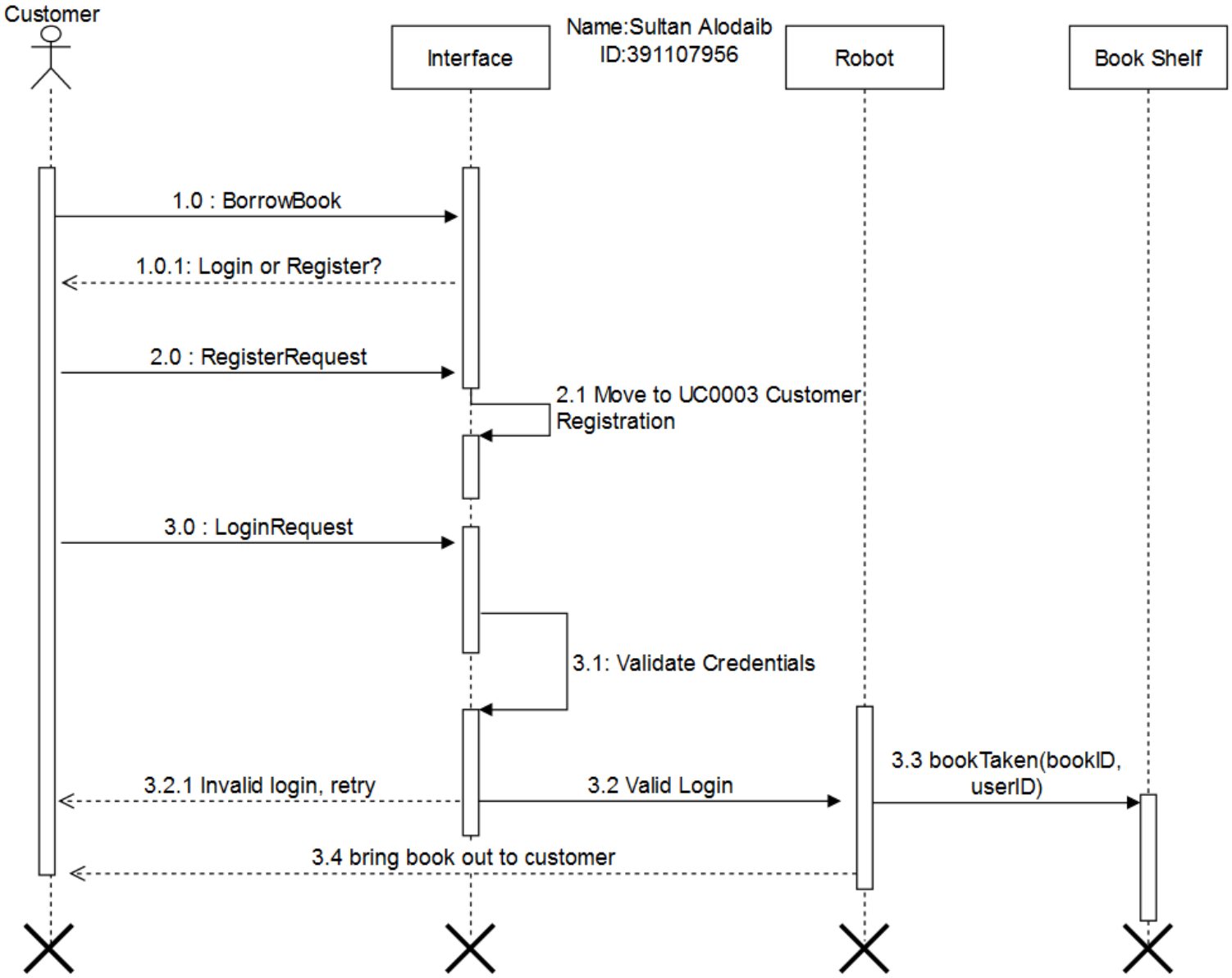
5.3 Sequence Diagrams:

5.3.1 Return book sequence:



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.

5.3.2 Borrow Sequence diagram



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 the customer

6 Human interface Design

6.1 About the GUI:

The GUI is outside of the scope of the system, the system provides an interface for external GUI design, which can be expanded upon for a full fledged Graphical User Interface.