Library Management Software Final Project

University of Houston - Downtown
Kinh Truong
Cooper Tran
Adrian Nguyen
Cerian Jaime
Nathan Nguyen

I. Introduction

For ages, libraries have been fundamental to the preservation and dissemination of knowledge, acting as a foundation for our society's growth in both education and culture. The fast development of digital technology, however, in recent years has fundamentally altered how libraries operate, creating new opportunities and difficulties. It is now simpler for users to obtain information and for librarians to manage their collections thanks to the development of electronic catalogs and online databases. Because of this, libraries have been able to keep up with the growing amount of material available and the changing demands of its users.

Despite these developments, a large number of neighborhood libraries in our city continue to maintain their collections using antiquated methods or manual procedures, which can result in inefficiencies and mistakes. These libraries might find it difficult to keep up with patron demand, which could result in lengthier wait times, mistakes with resource cataloging and circulation, and restricted access to crucial information. This could make it more difficult for libraries to play the vital function that they do in encouraging learning, education, and cross-cultural interaction. Given these difficulties, we have set out to develop a simple and efficient library management solution. The goal of our project is to create a system that makes it easier for patrons to access information and for librarians to manage collections by streamlining the classification, circulation, and retrieval of resources. We hope to make everyone's entire library experience by utilizing technology.

The traditional manual processes of library management are time-consuming, prone to errors, and difficult to scale. For example, manual tracking of books and materials can result in lost or misplaced items, and the manual process of tracking books and returns can be tedious and prone to errors. Additionally, many existing library management systems are either too complex or lack the necessary features to effectively manage modern libraries.

This project will develop a comprehensive library management system that addresses the aforementioned issues and provides a range of features to support modern libraries. The system will include modules for cataloging and tracking, managing orders and returns, and maintaining user accounts. The user interface will be intuitive and user-friendly, making it easy for admins, staff, and members to use the system.

We're building a library management system that's loaded with functions that are essential for effective and convenient library operations. It will first feature a comprehensive catalog of books, periodicals, and other resources with details on each item and the option to search and

categorize the content. Customers will find it easier to obtain information and find what they need as a result. Second, the system will provide a user account management tool that will let user login, make any personal profile changes, and search the catalog. A returns management tool will also be available, making it simpler for staff to manage the collection by keeping track of renewal possibilities and due dates. Finally, an admin interface with capabilities for handling user accounts, the entirety of the catalog, and sales report generation will be available to admins. With the help of these features, the library management system will run without manual errors, be simple to use, and efficiently meet the demands of libraries and their users.

In conclusion, our library management system promises to revolutionize library operations. It streamlines processes, reduces errors, and improves the user experience for both librarians and members. By incorporating modern technology, we aim to create a system that is both efficient and user-friendly, supporting libraries in their mission to provide access to knowledge and learning.

Glossary of Terms:

Catalog: A comprehensive list of books, periodicals, and other materials held by a library, including details such as title, author, and publication information.

Rent: Borrowing a library item for a specified time period, with the intention of returning it later. In a library management system, refers to checking out, tracking due dates, and monitoring renewals.

User Account Management: The process of managing patrons' library accounts, including details such as borrowing history, fines, and borrowing limits.

ISBN: Unique identifier for international books used for library cataloging and tracking

Administrative Interface: The interface that library staff use to manage the library's collections, user accounts, and system settings.

Reporting: The generation of detailed reports and statistics on library operations, including circulation data, inventory, and user activity.

II. Functional Requirements

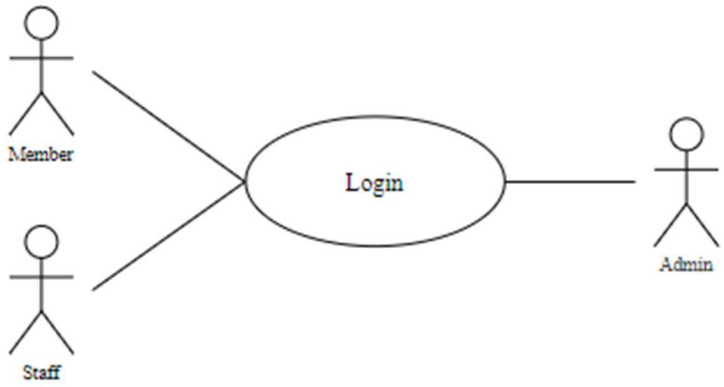
1. Software should allow members, staff, and admin to search for books.
2. Software should allow members, staff, and admin to login into their accounts.
3. Software should allow members, staff, and admin to edit personal profiles and change passwords.
4. Software should allow members, staff, and admin to view orders.
5. Software should allow staff and admin to manage orders and create orders.
6. Software should allow staff and admin to manage books.
7. Software should allow staff and admin to manage members, register and edit a member's profile.
8. Software should allow the admin to check orders.
9. Software should allow the admin to generate order reports.
10. Software should allow the admin to manage staff, register staff, and edit staff profiles.

III. Use Cases

Add New Members Use Case

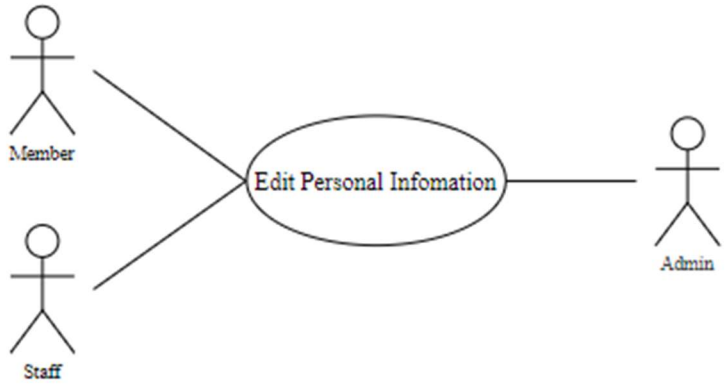
Brief Description: Admins can click "Add Member" and “Add Staff” button and will be prompted to register in the library system, following this, the admin enters the member and staff‘s personal information and enters their email and password to create the new account that the member can log into.

Login Use Case



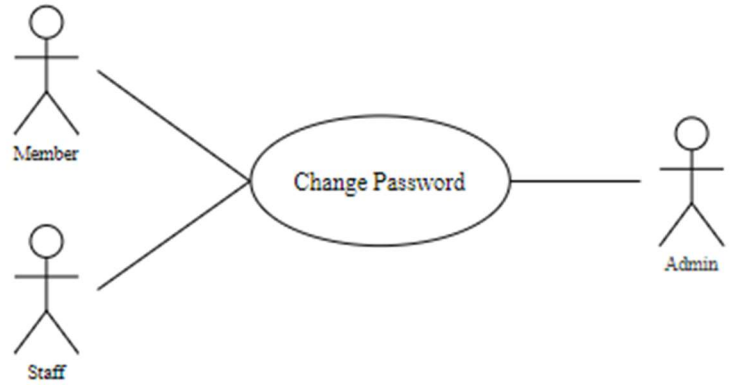
Brief Description: Clicking the "Login" button on the home page initiates the process of logging in as a member, staff, or admin. Upon clicking the login button, the member is directed to a form where they must enter their username and password. If the credentials are valid, the system logs the member in. Otherwise, the system displays an error message and prompts the member to try again.

Edit Personal Information Use Case



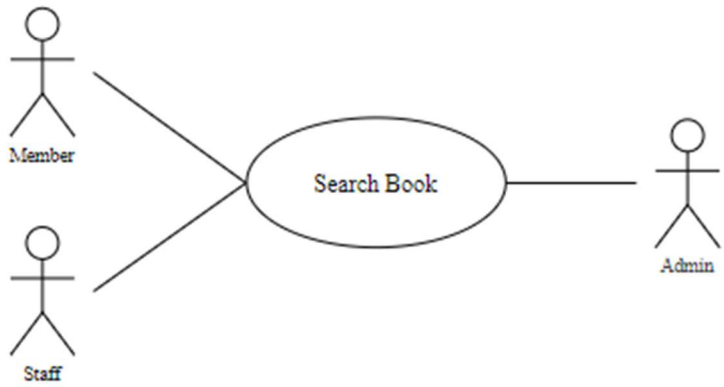
Brief Description: From the main menu, members, staff, and admin can select Edit Personal Profile to edit a member's personal information and account information. The system displays the current form for updating personal information. After making any necessary changes, the member clicks the "Submit" button to save the changes. It then displays a confirmation message informing the member that their information has been updated.

Change Password Use Case



Brief Description: From the main menu, members, staff, and admin can select Change Password to edit the password and click the "Submit" button to save the changes. It then displays a confirmation message informing the member that the password has been updated.

Search Book Use Case

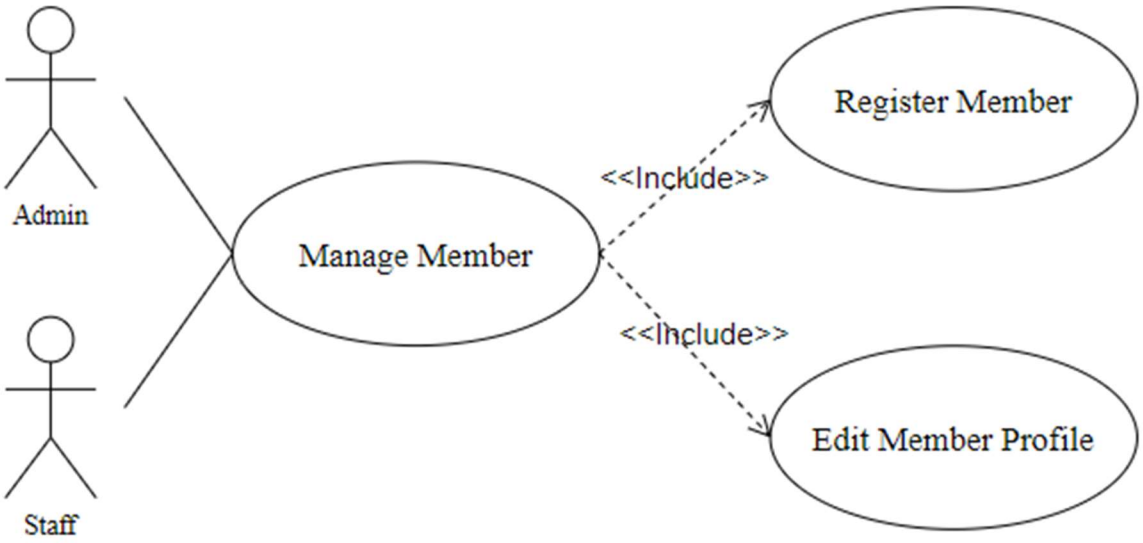


Brief Description: Members, staff, and admin begin the book search process by selecting the option to search for a book from the main menu. As soon as the search criteria, being the title or ISBN, the system displays the book search form. As soon as the "Search" button is clicked, the system displays a list of books that match his or her search criteria. In addition, they can view additional information about each book, such as its availability, and choose whether to check it out or add it to their wish list.

View Order Use Case

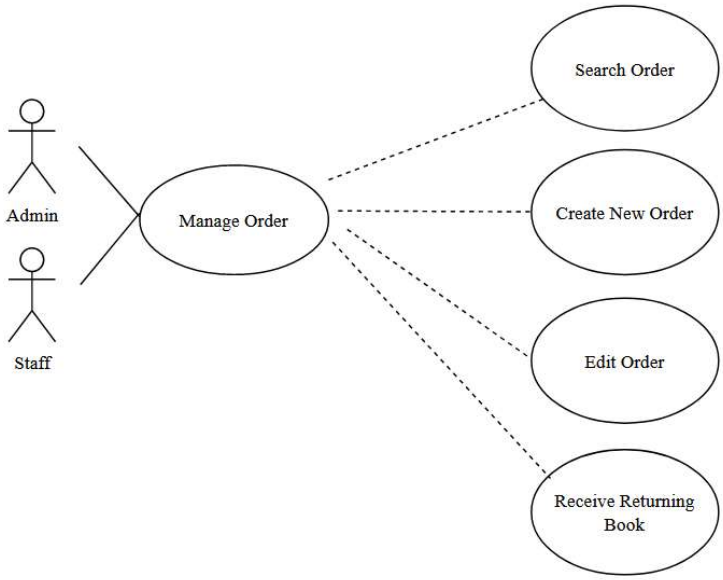
Brief Description: By selecting "View Order" from the main menu, members can view their current orders. Members are provided with a list of their orders, along with information regarding the book title, ISBN, and status.

Manage Member Use Case



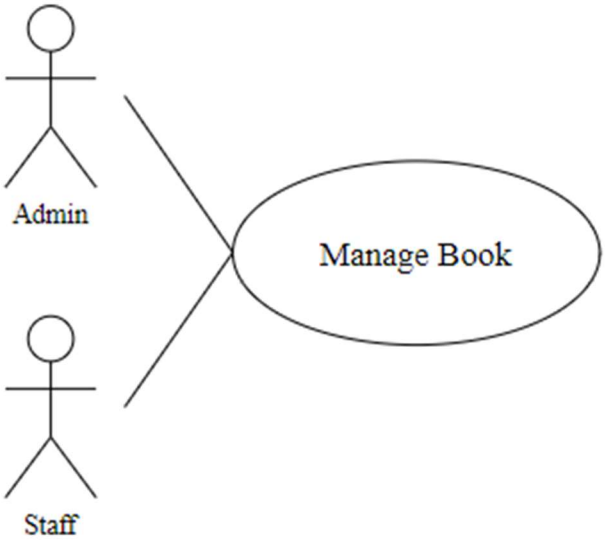
Brief Description: Staff and admin can Manage Member by Registering Member and Editing Member Profiles. New members may be registered by any staff and admin instead of members by completing the registration form and submitting it. When the information is verified, a new account will be created if it is valid. Otherwise, the system will prompt the staff to correct any errors. Staff and Admin can select a member's profile for editing, make the necessary changes, and submit the update, which is then confirmed by the system.

Manage Order Use Case



Brief Description: From the main menu, staff and admin can manage orders by selecting the option then a list of current orders appears. The user may view details, renew orders, check in books, or modify information. As soon as changes are made, a confirmation message is displayed, and the changes are saved. They can also view orders through View Order.

Manage Book Use Case

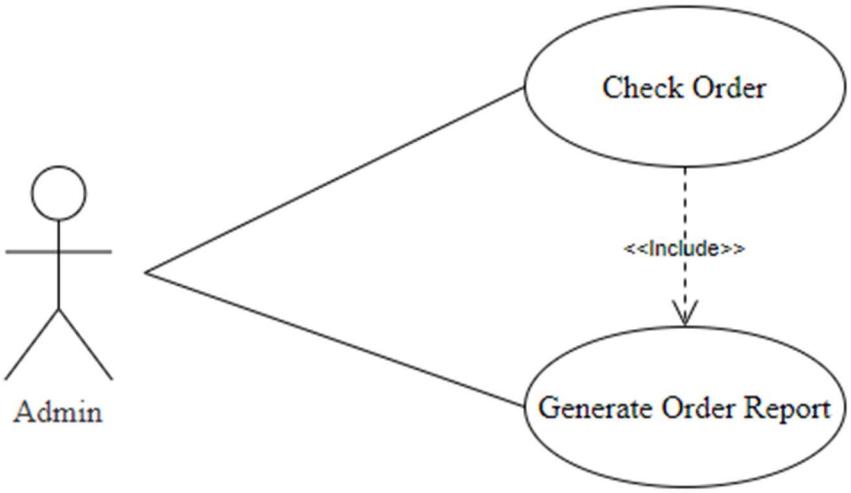


Brief Description: Upon selecting the "Manage Books" option from the main menu, a menu will show, allowing the admins and staff to Insert, Edit, Search, or Delete a book. A confirmation message appears after the changes have been saved by clicking the "Submit" button.

Manage Staff Use Case

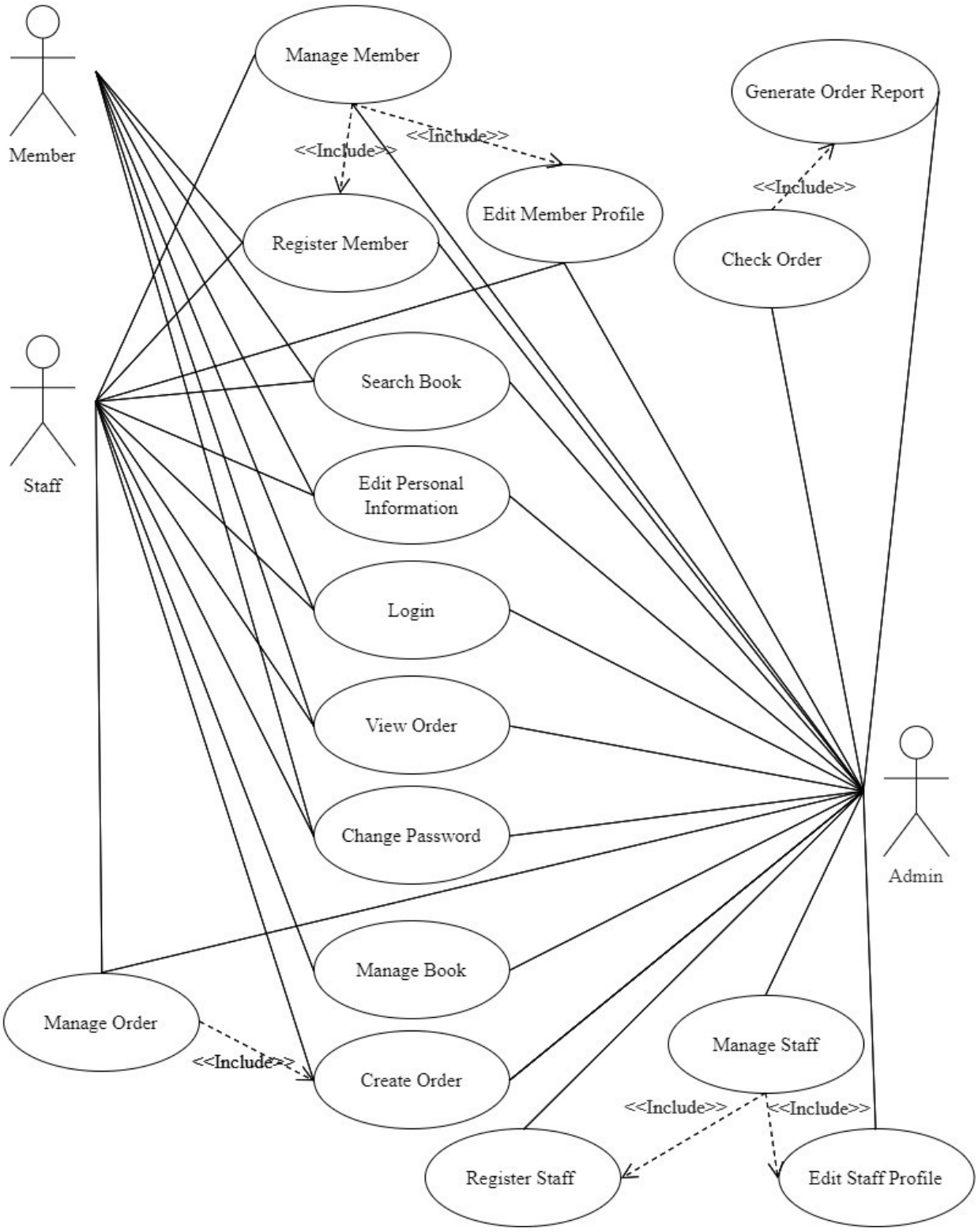
Brief Description: The admin can Manage Staff. The admin selects the "Register/Create Staff" option on the main menu and enters basic information such as name, email address, username, and password then clicks "Submit". The system then creates a new account if valid. Admin also has the ability to also add or select a member's profile for editing, make the necessary changes, and submit the update, which is then confirmed by the system.

Check Order & Generate Order Report Use Case

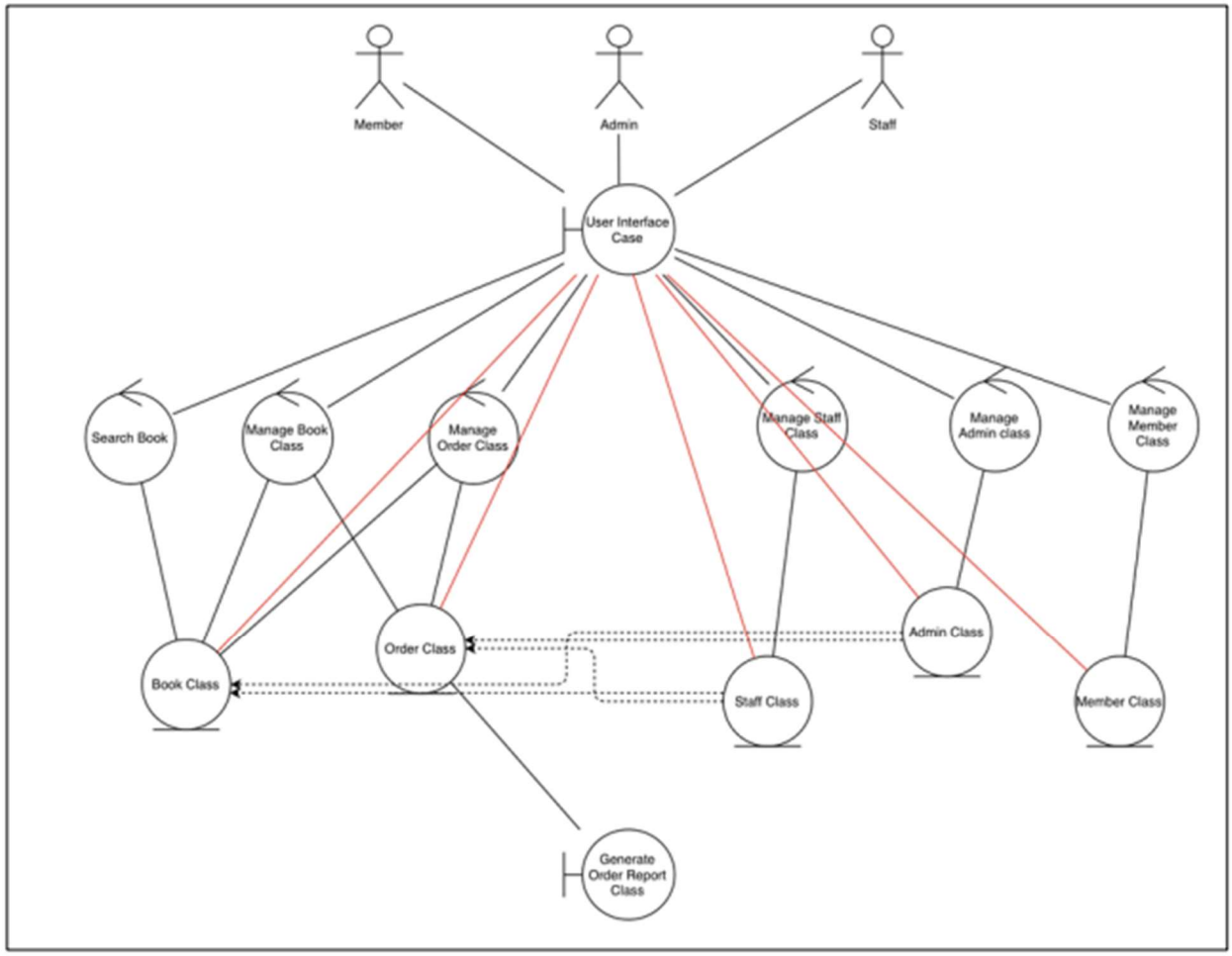


Brief Description: From the main menu, the admin can view orders. An administrator can also generate a sales report in this menu by selecting the option from the main menu, selecting the desired time period and format, and clicking the "Generate" button. This report will be created by the system and displayed for the administrator to view.

Library Management System



IV. Class Diagram



Classes with Attributes

Entity Classes:

<<entity class>> **Order Class**
- orderID - staffID - memberID - ISBN - rentDate - dueDate - returnDate - status

<<entity class>> **Book Class**
- title - author - ISBN - publisher - availability - shelf

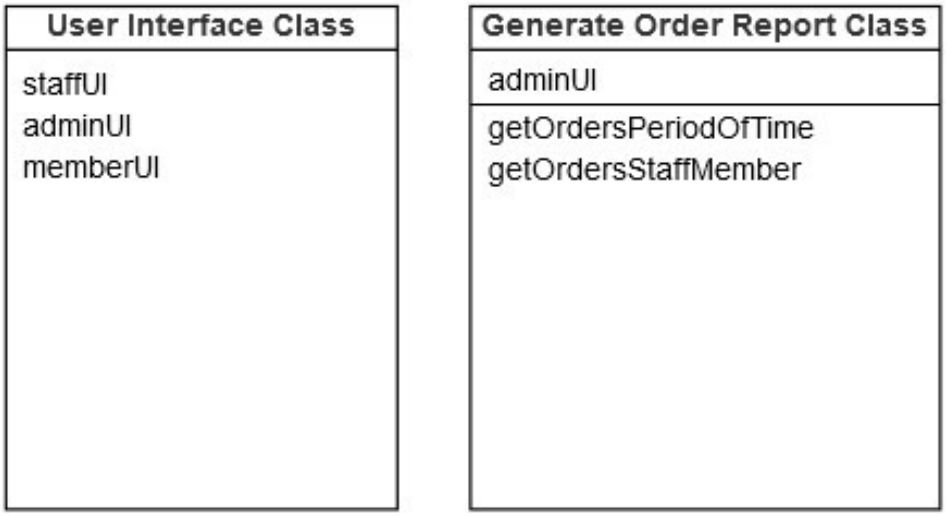
<<entity class>> **Member Class**
- memberID - fName - lName - email - password - status

<<entity class>> **Staff Class**
- staffID - fName - lName - email - password - status

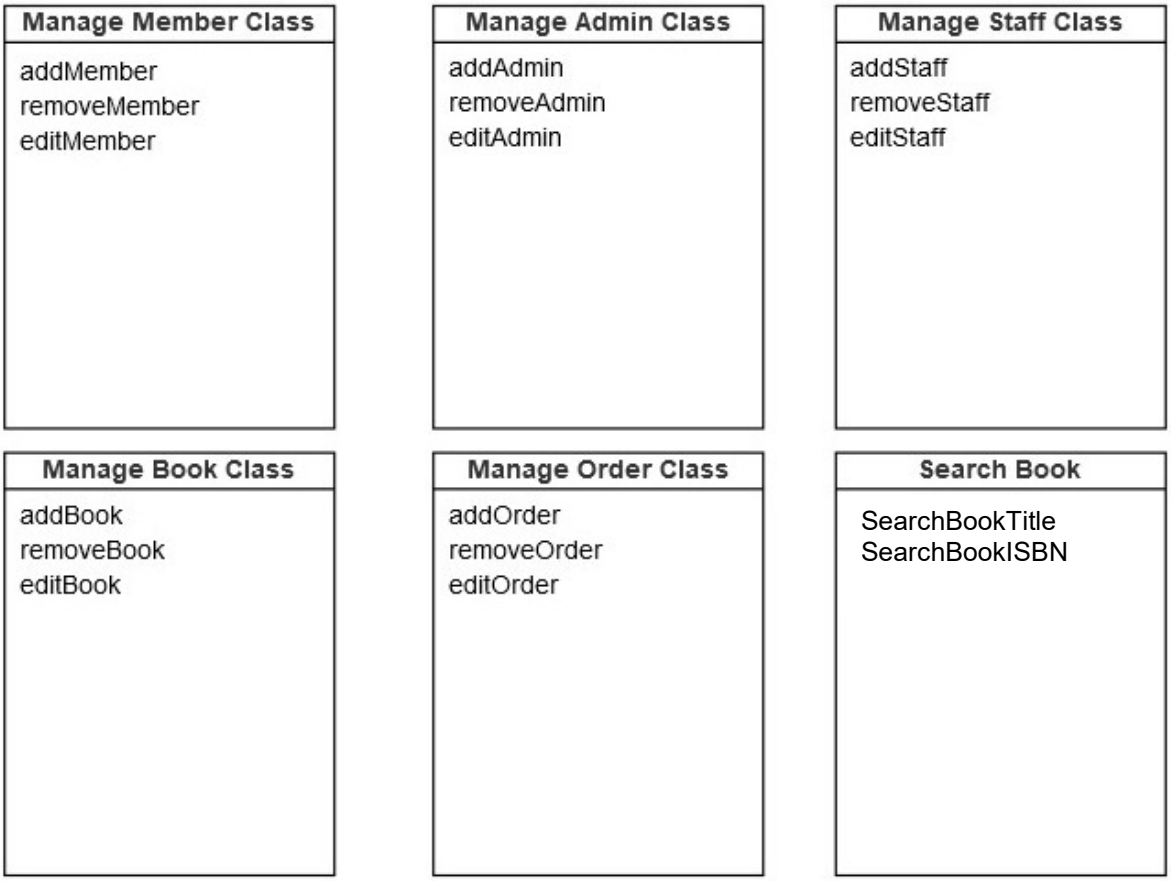
<<entity class>> **Admin Class**
- adminID - fName - lName - email - password - status

Originally the information for staff, members, and admins would include the address and phone number, but was changed to simplify the program to the parts that would be used more often like name, ID number, email, and password.

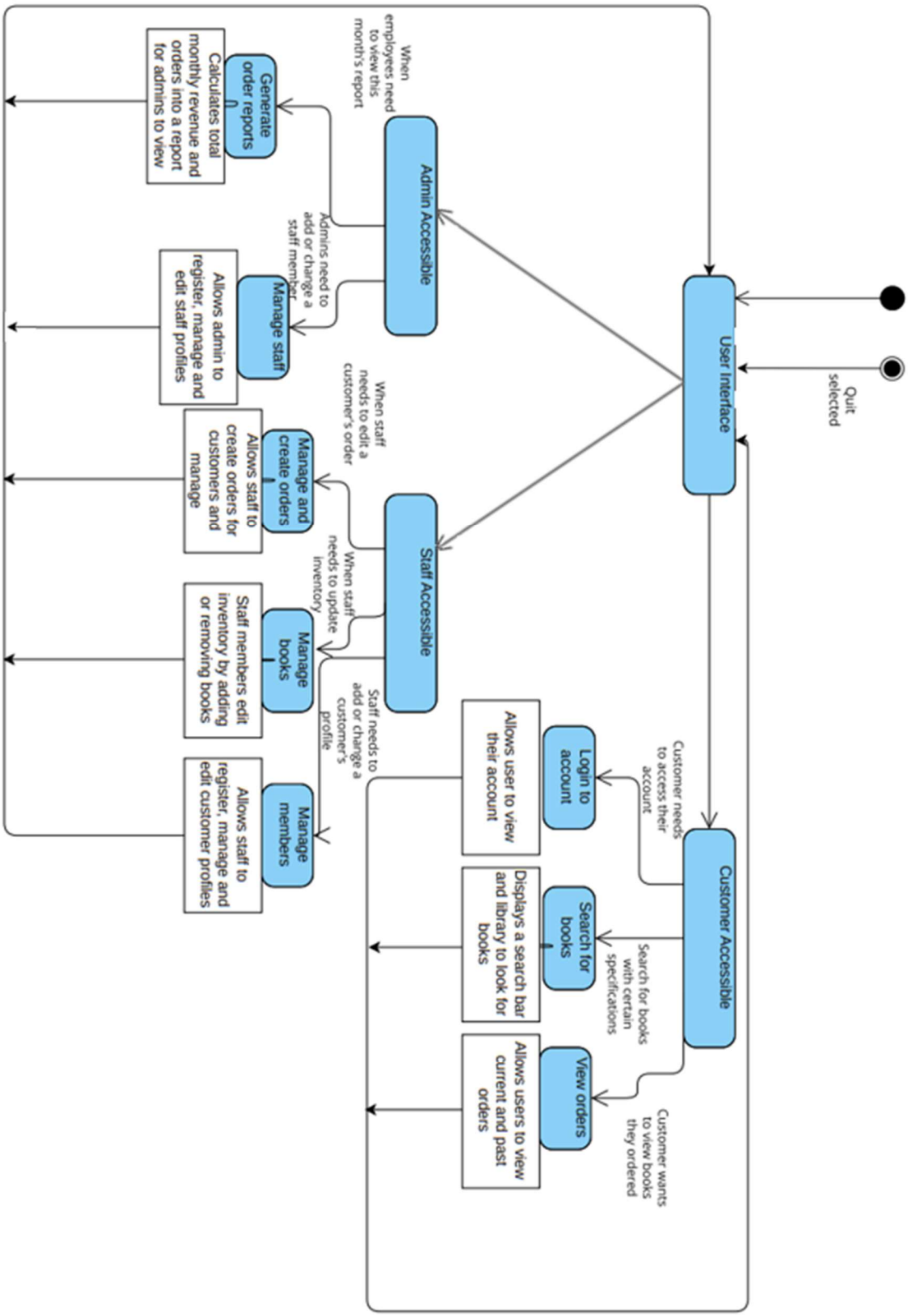
Boundary Classes:



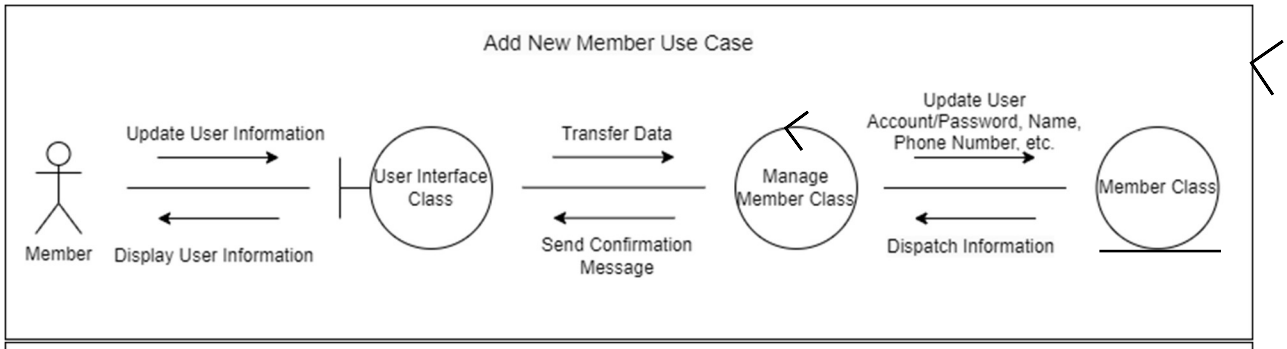
Control Classes:

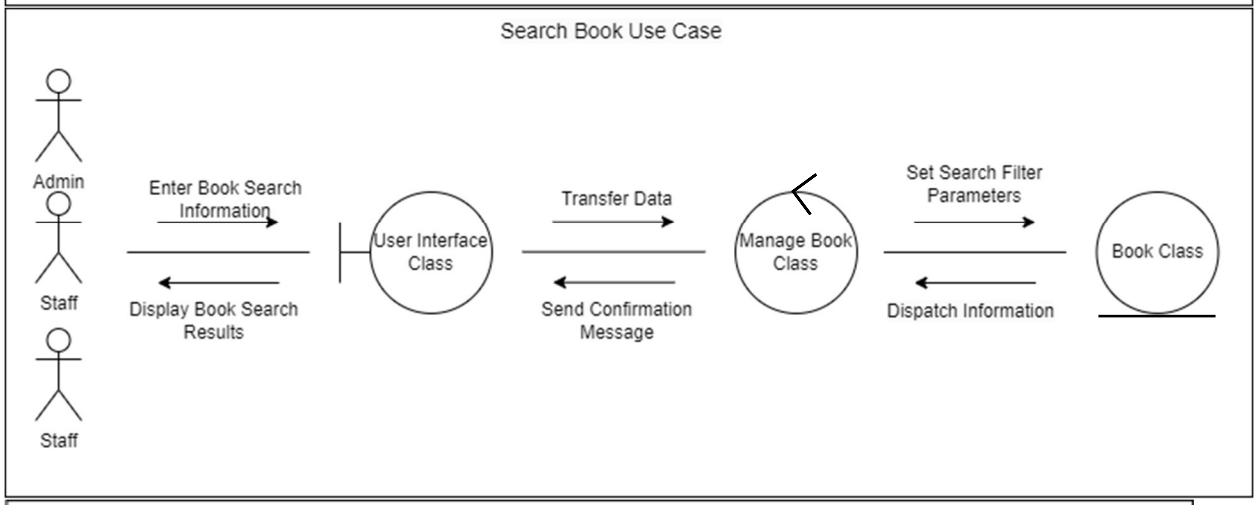


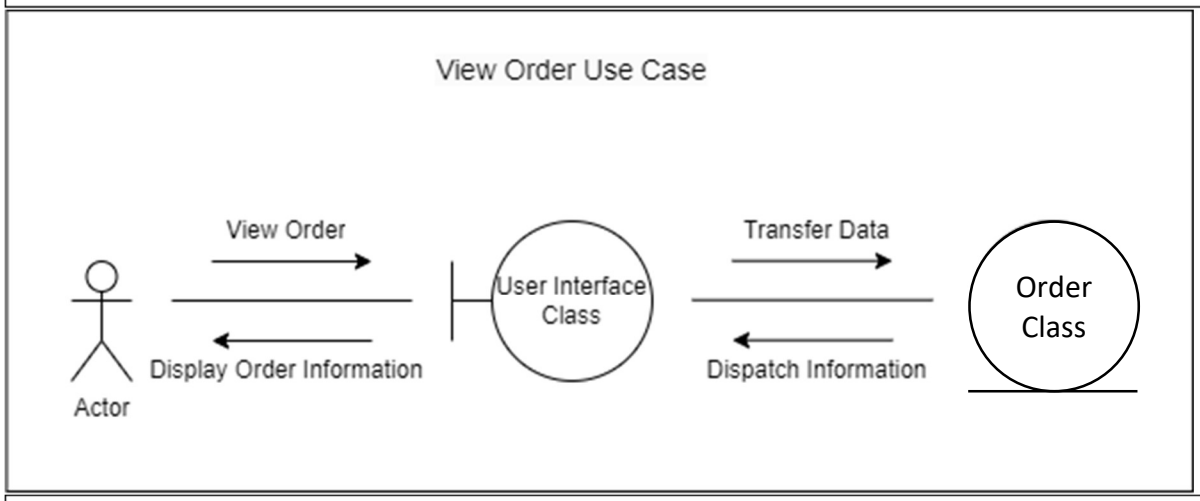
V. State Chart

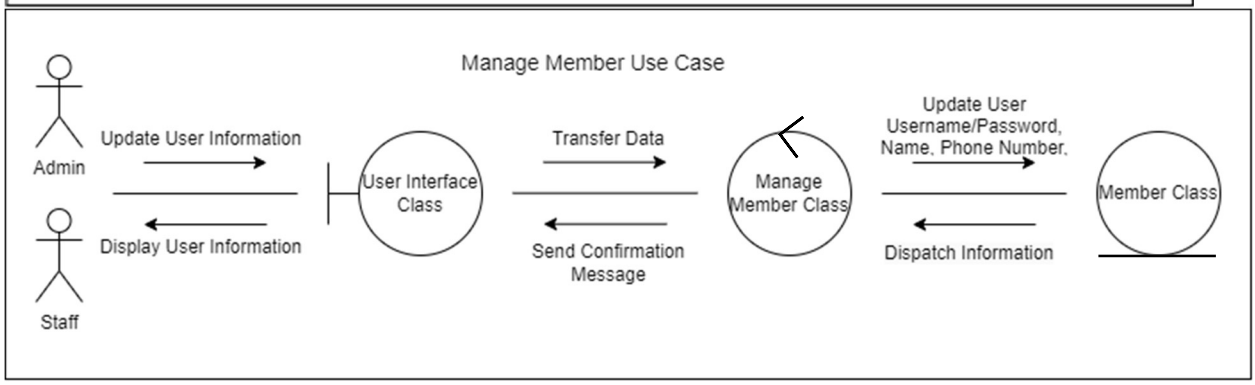


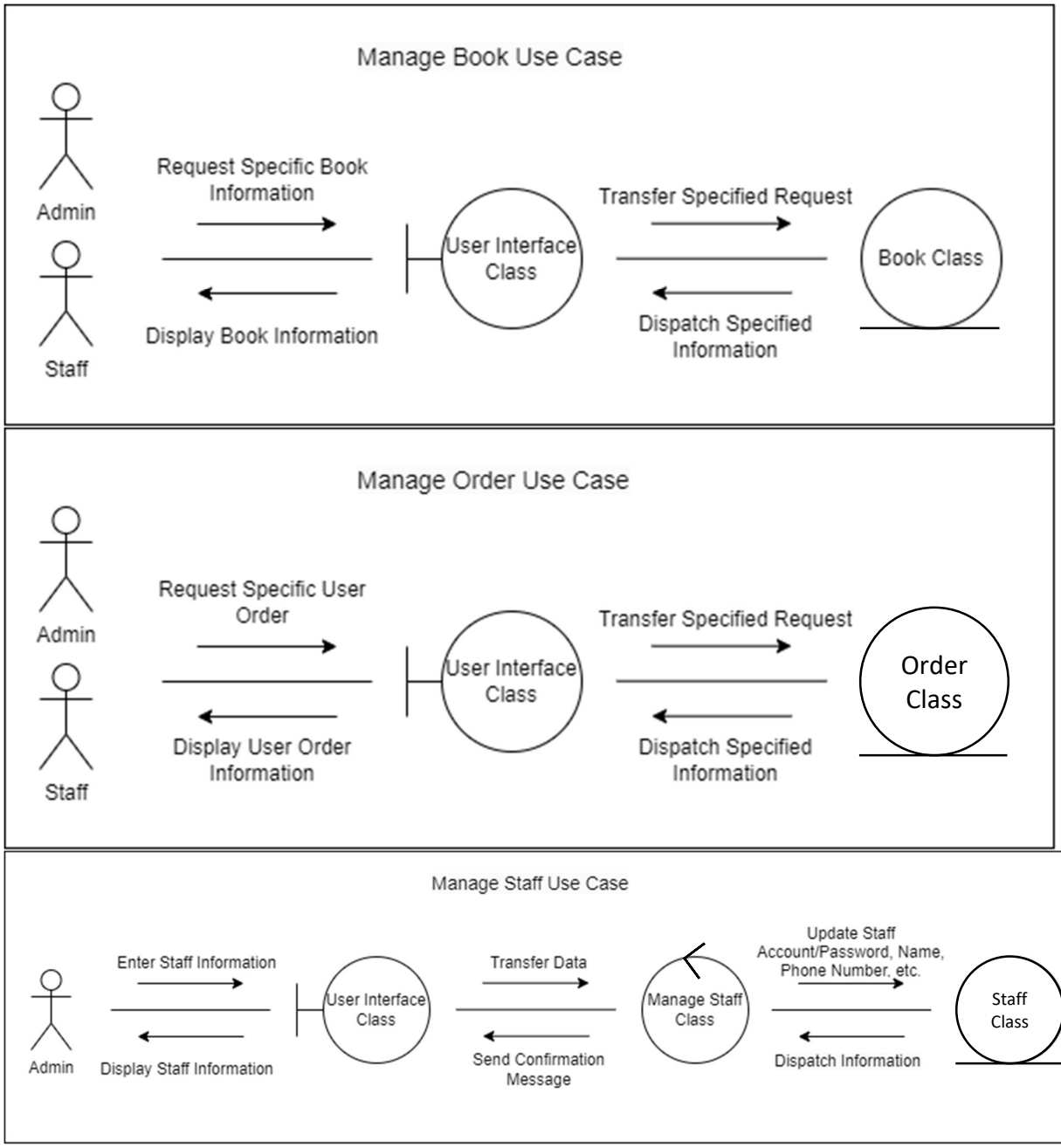
VI. Use Case Realization

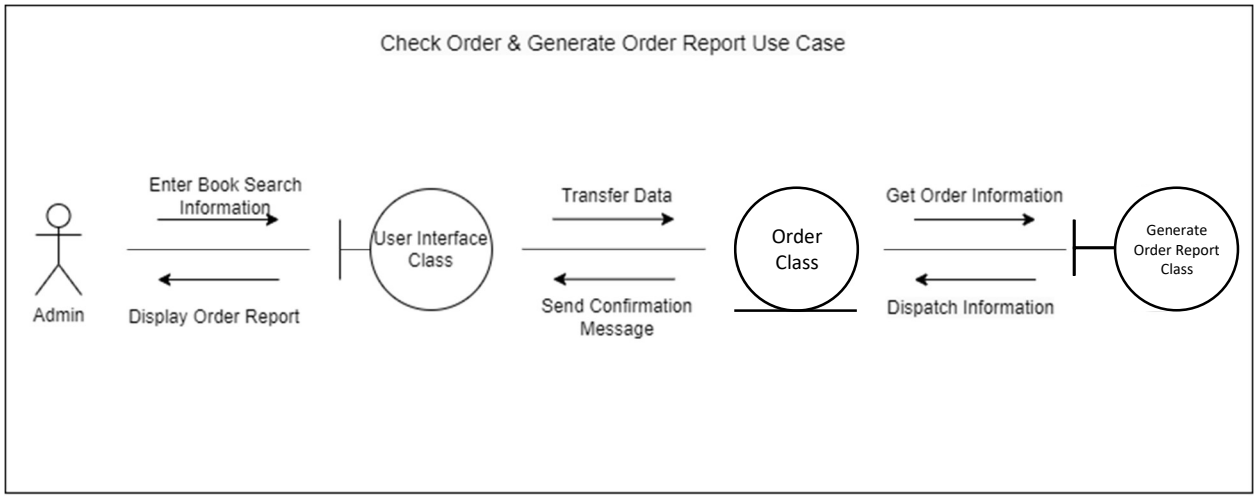

Add New Member Use Case
Member
Update User Information
Display User Information
User Interface Class
Transfer Data
Send Confirmation Message
Manage Member Class
Update User Account/Password, Name, Phone Number, etc.
Dispatch Information
Member Class


Search Book Use Case
Admin
Staff
Staff
Enter Book Search Information
Display Book Search Results
User Interface Class
Transfer Data
Send Confirmation Message
Manage Book Class
Set Search Filter Parameters
Dispatch Information
Book Class


View Order Use Case
Actor
View Order
Display Order Information
User Interface Class
Transfer Data
Dispatch Information
Order Class


Manage Member Use Case
Admin
Staff
Update User Information
Display User Information
User Interface Class
Transfer Data
Send Confirmation Message
Manage Member Class
Update User Username/Password, Name, Phone Number,
Dispatch Information
Member Class


Manage Book Use Case
Admin
Staff
Request Specific Book Information
Display Book Information
User Interface Class
Transfer Specified Request
Dispatch Specified Information
Book Class
Manage Order Use Case
Admin
Staff
Request Specific User Order
Display User Order Information
User Interface Class
Transfer Specified Request
Dispatch Specified Information
Order Class
Manage Staff Use Case
Admin
Enter Staff Information
Display Staff Information
User Interface Class
Transfer Data
Send Confirmation Message
Manage Staff Class
Update Staff Account/Password, Name, Phone Number, etc.
Dispatch Information
Staff Class



VII. Detailed Design

<<boundary class>> **User Interface Class**
+ <<static>> pressEnter() : void + <<static>> clearScreen() : void + <<static>> displayMainMenu() : void + <<static>> displayStaffMenu() : void + <<static>> displayMemberMenu() : void + <<static>> displayAdminMenu() : void + <<static>> displayReportMenu () : void + <<static>> displayBookMenu() : void + <<static>> getChar() : char + <<static>> getString() : string + <<static>> getInt() : int

void **pressEnter**() {
    Display “Press Enter to continue”;
    Await User Input;
}

void **clearScreen**() {
    Display “Clear Screen” over a popup;
    Await User Input
}

void **displayMainMenu**() {
    Display “Main Menu” over a popup;
    Await User Input;

```
}

void displayStaffMenu() {
        Display “User Menu” over a popup;
        Await User Input;
}

void displayMemberMenu() {
        Display “User Menu” over a popup;
        Await User Input;
}

void displayAdminMenu() {
        Display “User Menu” over a popup;
        Await User Input;
}

void displayReportMenu() {
        Display “Report Menu” over a popup;
        Await User Input;
}

void displayBookMenu() {
        Display “Book Menu” over a popup;
        Await User Input;
}

char getChar() {
        Await user input for storing char data
        return char;
}
string getString() {
        Await user input for storing string data
        return string;
}
int getInt() {
        Await user input for storing int data
        return int;
}
```

<<control class>> **Search Book Class**
+ getBookTitle () : string + getBookISBN () : string + displayBook () : void

```
string getBookTitle () {
        return SearchBookClass.Title;
}

string getBookISBN () {
        return searchBookClass.ISBN;
}

void displayBook () {
        for (all books) {
                If (available) {
                        display << searchBookClass.Title << " is available. ";
                }
                else {
                        display << searchBookClass.Title << " is not available. ";
                }
        }
}
```

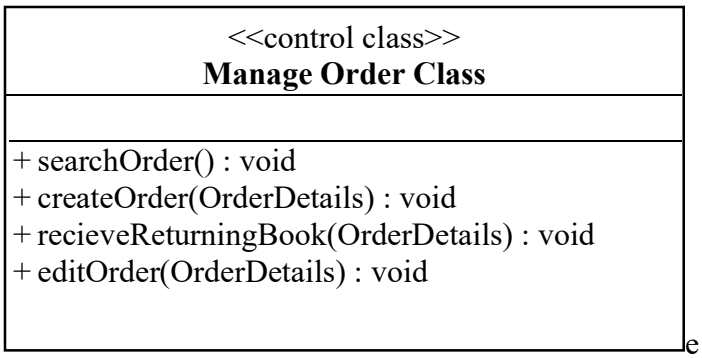
<<control class>> **Manage Book Class**
+ insertBook(BookDetails) : void + deleteBook(BookDetails) : void + editBook(BookDetails) : void

```
void insertBook(BookDetails) {
        insert newBook(BookDetails);
}

void deleteBook (BookDetails) {
        for (all books) {
                if (Book matches BookDetails) {
                        remove(book);
                        return;
                }
```

```
        }
}

void editBook (BookDetails) {
        for (all books) {
                if (Book matches BookDetails) {
                        write(BookDetails);
                        return;
                }
        }
}
```



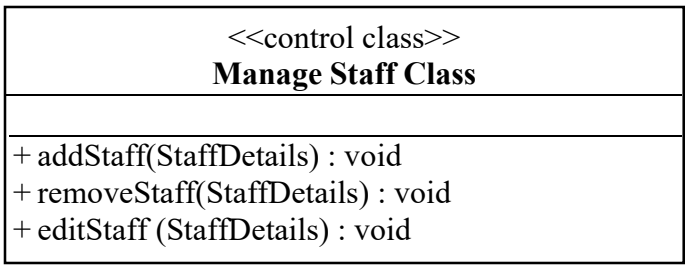
e

```
Void searchOrder(option) {
        If option 1{ searchOrderByISBN)();}
        If option 2 {searchOrderByMemberId();}
}

void createOrder(OrderDetails) {
        add newOrder(OrderDetails);
}

void recieveReturningBook (OrderDetails) {
        for (all orders) {
                if (Order matches OrderDetails) {
                        update(order);
                }
        }
}

void editOrder (OrderDetails) {
        for (all orders) {
                if (Order matches OrderDetails) {
                        write(OrderDetails);
                }
        }
}
```

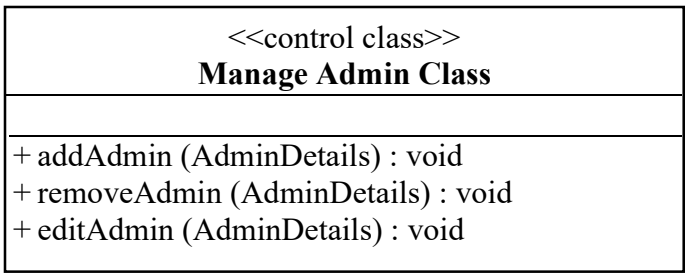


e

```
void addStaff(StaffDetails) {
        add newStaff(StaffDetails);
}

void removeStaff (StaffDetails) {
        for (all staffs) {
                if (Staff matches StaffDetails) {
                        remove(staff);
                        return;
                }
        }
}

void editStaff (StaffDetails) {
        for (all Staffs) {
                if (Staff matches StaffDetails) {
                        write(StaffDetails);
                        return;
                }
        }
}
```




```
void addAdmin (AdminDetails) {
        add newAdmin(AdminDetails);
}

void removeAdmin (AdminDetails) {
        for (all admins) {
                if (Admin matches AdminDetails) {
                        remove(admin);
                        return;
                }
        }
}

void editAdmin (AdminDetails) {
        for (all admins) {
                if (Admin matches AdminDetails) {
                        write(AdminDetails);
                        return;
                }
        }
}
```

<<control class>> **Manage Member Class**
+ searchMember (MemberDetails) : void + addMember (MemberDetails) : void + removeMember (MemberDetails) : void + editMember (MemberDetails) : void

```
Void searchMember (option) {
        if option 1{searchMemberByEmail(); }
        if option 2 {search MemberByName(); }
}

void addMember (MemberDetails) {
        add newMember(MemberDetails);
}

void removeMember (MemberDetails) {
        for (all members) {
                if (Member matches MemberDetails) {
                        remove(member);
                        return;
                }
        }
}

void editMember (MemberDetails) {
        for (all members) {
                if (Member matches MemberDetails) {
                        write(MemberDetails);
                        return;
                }
        }
}
```

<<entity class>> **Book Class**
- title : string - author : string - ISBN : string - publisher : string - availability : String - shelf : char
+ getTitle () : string + setTitle (n : string) : void + getAuthor () : string + setAuthor (n : string) : void + getISBN () : string + setISBN (n : string) : void + getPublisher () : string + setPublisher (n : string) : void + getAvailability (): string + setAvailability (n : string) : void + getShelf () : char + setShelf (n : char) : void + find (findBookID : string) : boolean + read (fileName : RandomAccessFile) : void + write (filename : RandomAccessFile) : void + deleteFile () : void + printFile () : void + <<static>> printAll () : void + updateTitle (): void + updateAuthor (): void + updateISBN (): void + updatepublisher (): void + updateAvailability (): void + updateShelf () : void

String **getTitle** () constant {
    Return BookClass.title;
}

Void **setTitle** (user_input) {
    BookClass.title = user_input;
}

String **getAuthor** () constant {
    Return BookClass.author;
}

```
Void setAuthor (user_input) {
        BookClass.author = user_input;
}

String getIsbn () constant {
        Return BookClass.isbn;
}

Void setISBN (user_input) {
        BookClass.ISBN = user_input;
}

String getPublisher () constant {
        Return BookClass.publisher;
}

Void setPublisher (user_input) {
        BookClass.publisher = user_input;
}

string getAvailability () constant {
        Return BookClass.availibility;
}

Void setAvailability (user_input) {
        BookClass.availibility = user_input;
}

char getShelf () constant {
        Return BookClass.shelf;
}

Void setShelf (user_input) {
        BookClass.shelf = user_input;
}

void read ( file_name ) {
        Open file_name;

        content = read next file line;
        BookClass.title

        content = read next file line;
        BookClass.author
```

```
        content = read next file line;
        BookClass.isbn

        content = read next file line;
        BookClass.publisher

        content = read next file line;
        BookClass.availibility

        content = read next file line;
        BookClass.shelf

        Close file;
}

void write ( file_name ) {
        Create file named file_name;

        Write << BookClass.title << "\n";
        Write << BookClass.author << "\n";
        Write << BookClass.isbn << "\n";
        Write << BookClass.publisher << "\n";
        Write << BookClass.availibilty << "\n";
        Write << BookClass.shelf << "\n";

        Close file;
}

void deleteFile (file_name) {
        Include system library;
        Delete file named file_name;
}

void printFile (file_name) {
        contents.Open(file_name);
        while(not end of file) {
                Display << contents << " ";
        }
        Close file;
}

void static printAll () {
        Display << "Book Information: " << "\n";
        Display << BookClass.title << "\n";
        Display << BookClass.author << "\n";
        Display << BookClass.isbn << "\n";
```

```
        Display << BookClass.publisher << "\n";
        Display << BookClass.availability << "\n";
        Display << BookClass.shelf << "\n";
}

void updateTitle () {
        Display << getTitle ();
        Get >> user_input;
        setTitle (user_input);
}

void updateAuthor () {
        Display << getAuthor ();
        Get >> user_input;
        setAuthor (user_input);
}

void updateIsbn () {
        Display << getIsbn ();
        Get >> user_input;
        setIsbn (user_input);
}

void updatePublisher () {
        Display << getPublisher ();
        Get >> user_input;
        setPublisher (user_input);
}

void updateAvailability () {
        Display << getAvailability ();
        Get >> user_input;
        setAvailability (user_input);
}

void updateShelf () {
        Display << getShelf ();
        Get >> user_input;
        setShelf (user_input);
}
```

<<entity class>>
Order Class

- orderID : string
- staffID : string
- memberID : string
- ISBN : string
- rentDate : String
- dueDate : String
- returnDate : String
- status : string

+ getOrderID (): string
+ setOrderID (n : string) : void
+ getStaffID () : string
+ setStaffID (n : string) : void
+ getMemberID () : string
+ setMemberID (n : string) : void
+ getISBN () : string
+ setISBN (n: string) : void
+ getRentDate () : String
+ setRentDate (n : String) : void
+ getDueDate () : String
+ setDueDate (n : String) : void
+ getReturnDate () : String
+ setReturnDate (n : String) : void
+ get status (n : string) : void
+ set status (n : string) : void
+ find (findOrderID : string) : boolean
+ read (fileName : RandomAccessFile) : void
+ write (filename : RandomAccessFile) : void
+ deleteFile () : void
+ printFile () : void
+ <<static>> printAll () : void
+ updateOrderID () : void
+ updateStaffID () : void
+ updateMemberID () : void
+ updateBookID () : void
+ updateRentDate () : void
+ updateDueDate () : void
+ updateReturnDate () : void
+ updateStatus () : void

```
String getOrderID () constant {
        Return OrderClass.oderID;
}
```

```
Void setOrderID (user_input) {
        OrderClass.orderID = user_input;
}

String getStaffID () constant {
        Return OrderClass.staffID;
}

Void setStaffID (user_input) {
        OrderClass.staffID = user_input;
}

String getMemberID () constant {
        Return OrderClass.memberID;
}

Void setMemberID (user_input) {
        OrderClass.memberID = user_input;
}

String ISBN () constant {
        Return OrderClass.ISBN;
}

Void ISBN (user_input) {
        OrderClass.ISBN = user_input;
}

String getRentDate () constant {
        Return OrderClass.rentDate;
}

Void setRentDate (user_input) {
        OrderClass.rentDate = user_input;
}

String getDueDate () constant {
        Return OrderClass.dueDate;
}

Void setDueDate (user_input) {
        OrderClass.dueDate = user_input;
}

String getReturnDate () constant {
        Return OrderClass.returnDate;
```

```
}

Void setReturnDate (user_input) {
        OrderClass.returnDate = user_input;
}

String getStatus () constant {
        Return OrderClass.status;
}

Void setStatus (user_input) {
        OrderClass.status = user_input;
}

void read ( file_name ) {
        Open file_name;

        content = read file line;
        OrderClass.orderID = content;

        content = read next file line;
        OrderClass.staffID

        content = read next file line;
        OrderClass.memberID

        content = read next file line;
        OrderClass.ISBN

        content = read next file line;
        OrderClass.rentDate

        content = read next file line;
        OrderClass.dueDate

        content = read next file line;
        OrderClass.returnDate

        Content = read next file line;
        OderClass.status

        Close file;
}

void write ( file_name ) {
        Create file named file_name;
```

```
        Write << OrderClass.orderID << "\n";
        Write << OrderClass.staffID << "\n";
        Write << OrderClass.memberID << "\n";
        Write << OrderClass.ISBN << "\n";
        Write << OrderClass.rentDate << "\n";
        Write << OrderClass.dueDate << "\n";
        Write << OrderClass.returnDate << "\n";
        Write << OrderClass.status << "\n";

        Close file;
}

void deleteFile (file_name) {
        Include system library;
        Delete file named file_name;
}

void printFile (file_name) {
        contents.Open(file_name);
        while(not end of file) {
                Display << contents << " ";
        }
        Close file;
}

void static printAll () {
        Display << OrderClass.orderID << "\n";
        Display << OrderClass.staffID << "\n";
        Display << OrderClass.memberID << "\n";
        Display << OrderClass.ISBN << "\n";
        Display << OrderClass.rentDate << "\n";
        Display << OrderClass.dueDate << "\n";
        Display << OrderClass.returnDate << "\n";
        Display << OrderClass.status << "\n";
}

void updateOrderID () {
        Display << getOrderID ();
        Get >> user_input;
        setOrderID (user_input);
}

void updateStaffID () {
        Display << getStaffID ();
        Get >> user_input;
```

```
        setStaffID (user_input);
}

void updateMemberID () {
        Display << getMemberID ();
        Get >> user_input;
        setMemberID (user_input);
}

void updateISBN () {
        Display << getISBN ();
        Get >> user_input;
        setISBN (user_input);
}

void updateRentDate () {
        Display << getRentDate ();
        Get >> user_input;
        setRentDate (user_input);
}

void updateDueDate () {
        Display << getDueDate ();
        Get >> user_input;
        setDueDate (user_input);
}

void updateReturnDate () {
        Display << getReturnDate ();
        Get >> user_input;
        setReturnDate (user_input);
}

Void upadteStatus () {
        Display << getStatus ();
        Get >> user_input;
        setStatus (user_input);
}
```

<<entity class>> **Member Class**
- memberID : int - fName : string - lName : string - email : string - password : string - status : string
+ getMemberID () : int + setMemberID (n : int) : void + getFName () : string + setFName (n : string) : void + getLName () : string + setLName (n : string) : void + getEmail () : string + setEmail (n : string) : void + getPassword () : string + setPasswoed (n : string) : void + getStatus (n : string) : void + setStatus (n : string) : void + find (findMemberID : int) : boolean + read (fileName : RandomAccessFile) : void + write (filename : RandomAccessFile) : void + deleteFile () : void + printFile () : void + <<static>> printAll () : void + updateMemberID () : void + updateFName () : void + updateLName () : void + updateEmail () : void + updatePassword () : void + updateStatus () : void

String **getMemberID** () constant {
    Return MemberClass.memberID;
}

Void **setMemberID** (user_input) {
    MemberClass.memberID = user_input;
}

String **getFName** () constant {
    Return MemberClass.fName;
}

```
Void setFName (user_input) {
        MemberClass.fName = user_input;
}

String getLName () constant {
        Return MemberClass.lName;
}

Void setLName (user_input) {
        MemberClass.lName = user_input;
}

String getEmail () constant {
        Return MemberClass.email;
}

Void setEmail (user_input) {
        MemberClass.email = user_input;
}

String getPassword () constant {
        Return MemberClass.password;
}

Void setPassword (user_input) {
        MemberClass.password = user_input;
}

String getStatus () constant {
        Return MemberClass.status;
}

Void setStatus (user_input) {
        MemberClass.status = user_input;
}


void read ( file_name ) {
        Open file_name;

        content = read file line;
        MemberClass.memberID = content;

        content = read next file line;
        MemberClass.fName
```

```
        content = read next file line;
        MemberClass.lName

        content = read next file line;
        MemberClass.email

        content = read next file line;
        MemberClass.password

        Content = read next file line;
        MemberClass.status

        Close file;
}

void write ( file_name ) {
        Create file named file_name;

        Write << MemberClass.memberID << "\n";
        Write << MemberClass.fName << "\n";
        Write << MemberClass.lName << "\n";
        Write << MemberClass.email << "\n";
        Write << MemberClass.password << "\n";
        Write << MemberClass.status << "\n";

        Close file;
}

void deleteFile (file_name) {
        Include system library;
        Delete file named file_name;
}

void printFile (file_name) {
        contents.Open(file_name);
        while(not end of file) {
                Display << contents << " ";
        }
        Close file;
}

void static printAll () {
        Display << MemberClass.memberID << "\n";
        Display << MemberClass.fName << "\n";
        Display << MemberClass.lName << "\n";
        Display << MemberClass.email << "\n";
```

```
        Display << MemberClass.password << "\n";
        Display << MemberClass.status << "\n";
}

void updateMemberID () {
        Display << getMemberID ();
        Get >> user_input;
        setMemberID (user_input);
}

void updateFName () {
        Display << getFName ();
        Get >> user_input;
        setFName (user_input);
}

void updateLName () {
        Display << getLName ();
        Get >> user_input;
        setLName (user_input);
}

void updateEmail () {
        Display << getEmail ();
        Get >> user_input;
        setEmail (user_input);
}

void updatePassword () {
        Display << getPassword ();
        Get >> user_input;
        setPassword (user_input);
}

void updateStatus () {
        Display << getStatus ();
        Get >> user_input;
        setStatus (user_input);
}
```

<<entity class>> **Staff Class**
- staffID : int - fName : string - lName : string - emai : string - password : string - status : string
+ getStaffID () : int + setStaffID (n : int) : void + getFName () : string + setFName (n : string) : void + getLName () : string + setLName (n : string) : void + getEmail () : string + setEmail (n : string) : void + getPassword () : string + setPasswoed (n : string) : void + getStatus () : string + setStatus (n : string) : void + find (findStaffID : int) : boolean + read (fileName : RandomAccessFile) : void + write (filename : RandomAccessFile) : void + deleteFile () : void + printFile () : void + <<static>> printAll () : void + updateStaffID () : void + updateFName () : void + updateLName () : void + updateEmail () : void + updatePassword () : void + updateStatus () : void

```
String getStaffID () constant {
        Return StaffClass.StaffID;
}

Void setStaffID (user_input) {
        StaffClass.StaffID = user_input;
}

String getFName () constant {
        Return StaffClass.fName;
}
```

```
Void setFName (user_input) {
        StaffClass.fName = user_input;
}

String getLName () constant {
        Return StaffClass.lName;
}

Void setLName (user_input) {
        StaffClass.lName = user_input;
}

String getEmail () constant {
        Return StaffClass.email;
}

Void setEmail (user_input) {
        StaffClass.email = user_input;
}

String getPassword () constant {
        Return StaffClass.password;
}

Void setPassword (user_input) {
        StaffClass.password = user_input;
}

String getStatus () constant {
        Return StaffClass.status;
}

Void setStatus (user_input) {
        StaffClass.status = user_input;
}


void read ( file_name ) {
        Open file_name;

        content = read file line;
        StaffClass.staffID = content;

        content = read next file line;
        StaffClass.fName
```

```
        content = read next file line;
        StaffClass.lName

        content = read next file line;
        StaffClass.email

        content = read next file line;
        StaffClass.password

        content = read next file line;
        StaffClass.status

        Close file;
}

void write ( file_name ) {
        Create file named file_name;

        Write << StaffClass.staffID << "\n";
        Write << StaffClass.fName << "\n";
        Write << StaffClass.lName << "\n";
        Write << StaffClass.email << "\n";
        Write << StaffClass.password << "\n";
        Write << StaffClass.status << "\n";

        Close file;
}

void deleteFile (file_name) {
        Include system library;
        Delete file named file_name;
}

void printFile (file_name) {
        contents.Open(file_name);
        while(not end of file) {
                Display << contents << " ";
        }
        Close file;
}

void static printAll () {
        Display << StaffClass.staffID << "\n";
        Display << StaffClass.fName << "\n";
        Display << StaffClass.lName << "\n";
        Display << StaffClass.email << "\n";
```

```
        Display << StaffClass.password << "\n";
        Display << StaffClass.status << "\n";
}

void updateStaffID () {
        Display << getStaffID ();
        Get >> user_input;
        setStaffID (user_input);
}

void updateFName () {
        Display << getFName ();
        Get >> user_input;
        setFName (user_input);
}

void updateLName () {
        Display << getLName ();
        Get >> user_input;
        setLName (user_input);
}

void updateEmail () {
        Display << getEmail ();
        Get >> user_input;
        setEmail (user_input);
}

void updatePassword () {
        Display << getPassword ();
        Get >> user_input;
        setPassword (user_input);
}

void updateStatus () {
        Display << getStatus ();
        Get >> user_input;
        setStatus (user_input);
}
```

<<entity class>> **Admin Class**
- adminID : int - fName : string - lName : string - emai : string - password : string - status : string
+ getAdminID () : int + setAdminID (n : int) : void + getFName () : string + setFName (n : string) : void + getLName () : string + setLName (n : string) : void + getEmail () : string + setEmail (n : string) : void + getPassword () : string + setPasswoed (n : string) : void + getStatus () : string + setStatus (n : string) : void + find (findAdminID : int) : boolean + read (fileName : RandomAccessFile) : void + write (filename : RandomAccessFile) : void + deleteFile () : void + printFile () : void + <<static>> printAll () : void + updateAdminID () : void + updateFName () : void + updateLName () : void + updateEmail () : void + updatePassword () : void + updateStatus () : void

String **getAdminID** () constant {
    Return AdminClass.adminID;
}

Void **setAdminID** (user_input) {
    AdminClass.adminID = user_input;
}

String **getFName** () constant {
    Return AdminClass.fName;
}

```
Void setFName (user_input) {
        AdminClass.fName = user_input;
}

String getLName () constant {
        Return AdminClass.lName;
}

Void setLName (user_input) {
        AdminClass.lName = user_input;
}

String getEmail () constant {
        Return AdminClass.email;
}

Void setEmail (user_input) {
        AdminClass.email = user_input;
}

String getPassword () constant {
        Return AdminClass.password;
}

Void setPassword (user_input) {
        AdminClass.password = user_input;
}

String getStatus () constant {
        Return AdminClass.status;
}

Void setStatus (user_input) {
        AdminClass.status = user_input;
}

void read ( file_name ) {
        Open file_name;

        content = read file line;
        AdminClass.adminID = content;

        content = read next file line;
        AdminClass.fName

        content = read next file line;
```

```
        AdminClass.lName

        content = read next file line;
        AdminClass.email

        content = read next file line;
        AdminClass.password

        content = read next file line;
        AdminClass.status

        Close file;
}

void write ( file_name ) {
        Create file named file_name;

        Write << AdminClass.adminID << "\n";
        Write << AdminClass.fName << "\n";
        Write << AdminClass.lName << "\n";
        Write << AdminClass.email << "\n";
        Write << AdminClass.password << "\n";
        Write << AdminClass.status << "\n";

        Close file;
}

void deleteFile (file_name) {
        Include system library;
        Delete file named file_name;
}

void printFile (file_name) {
        contents.Open(file_name);
        while(not end of file) {
                Display << contents << " ";
        }
        Close file;
}

void static printAll () {
        Display << AdminClass.adminID << "\n";
        Display << AdminClass.fName << "\n";
        Display << AdminClass.lName << "\n";
        Display << AdminClass.email << "\n";
        Display << AdminClass.password << "\n";
```

```
        Display << AdminClass.status << "\n";
}

void updateAdminID () {
        Display << getAdminID ();
        Get >> user_input;
        setAdminID (user_input);
}

void updateFName () {
        Display << getFName ();
        Get >> user_input;
        setFName (user_input);
}

void updateLName () {
        Display << getLName ();
        Get >> user_input;
        setLName (user_input);
}

void updateEmail () {
        Display << getEmail ();
        Get >> user_input;
        setEmail (user_input);
}

void updatePassword () {
        Display << getPassword ();
        Get >> user_input;
        setPassword (user_input);
}

void updateStatus () {
        Display << getStatus ();
        Get >> user_input;
        setStatus (user_input);
}
```

<<boundary class>> **Generate Order Report Class**
+ <<static>> printReport() : void

```
Void printReport () {
```

```
        Display << getStaffID(), getMemberID, getAvailableBooks(), getBookLoans(),
        getOrderLoans(), getOverdueOrders();
}
```

VIII. Implementation

Selection of Programming Language and IDE

For this project, we chose to use Visual Studio 2019 and Python along with MySQL to use as our database management system that we would host locally. The database we set up in MySQL would have tables that would hold the data for all necessary classes. We chose to use Python because of our familiarity with it and how easier it is to read. MySQL was chosen because of its popular usage in web development and databases.

Programming Practices

Because of the integration of MySQL into our project, we have comments that would provide the information and context needed to understand the logic of the code. We also wanted to make sure that functions that use conditional statements to not be too extensive as that would make the code unreadable and hard to maintain. We also made sure that the variables had names that were simple and descriptive enough for one to understand.

Reuse and Portability

As for reusing code, the most prominent example of that would be the functions used to show the user the menu and to navigate through it, which is something that we have used in previous projects. The most useful code that we could reuse in the future is the functions used to input emails and passwords in the menu to create an account. It would then be stored in the table we created in MySQL. Now when logging in, the application can reference that table in the database and grant access to the account. What makes this code useful is that its logic is also used to input and store the information for the books as well. This allows the code logic to be used for various applications extending past login information.

As for portability, considering the use of MySQL and Python, this application should be able to run on all major operating systems.

Version Control System

As for the version control system that we used, Git was the first pick because it was free and because of how easy it was to share code. Git is open source, which simplifies working on

the same project at the same time, making it the most useful feature. We do not think that there are any features that should be added

IX. Test Cases and Results

Use Case: Search Book
Expect: After inserting the title or ISBN to search, books' information will be displayed
Result: books' information is displayed after searching

```
Search book title: Harry Porter
('9780747532743', 'Harry Potter and the Chamber of Secrets', 'J.K. Rowling', 'B
loomsbury Publishing', 'On Loan', 'A')
Press ENTER to back to Search MENU
```

```
Search book ISBN: 9780545010221
('9780545010221', 'The Hunger Games', 'Suzanne Collins', 'Scholastic Press', 'A
vailable', 'C')
Press ENTER to back to Search MENU
```

Use Case: View Your Order
Expect: all orders of the member will be displayed
Result: all orders is displayed

```
|OrderID ||     ISBN      ||  Status  |
('000002', '9780747532743', 'Overdue')
Enter orderID to view detail
(Enter '-1' to LOG OUT)
ENTER orderID:
```

Use Case: View Order Detail
Expect: After entering the order ID, details of the order will be displayed
Result: The details of the order is displayed

```
OrderID: 000002 Status: Overdue
StaffID: ST001  Name: Adrian Nguyen
MemberID: MB001 Name: Alice Jones
Book ISBN: 9780747532743
Book Title: Harry Potter and the Chamber of Secrets
Book Author: J.K. Rowling
Rent Date: 2023-04-25 13:45:00       Due Date: 2023-04-30
Return Date: None
(Press ENTER to return)
```

Use Case: Edit Personal Information
Expect: Allow the user to edit the name, email address, and password and update into database
Result: The user is able to change their name, email, and password

```
ID: MB001
Name: Alice Jones
Email: alice.jones@example.com
Password: **********
Menu:
1. Edit Name
2. Edit Email
3. Edit Password
(Enter '-1' to Back)
ENTER your action:
```

```
ID: MB001
Name: Alex Johnson
Email: alex.john@gmail.com
Password: **********
Menu:
1. Edit Name
2. Edit Email
3. Edit Password
(Enter '-1' to Back)
ENTER your action:
```

Use Case: Insert New Book
Expect: Allow Staff and Admin to insert new books into the database
Result: Staff and Admin can insert new books into the database

```
New Book Information
ISBN: 9780062316110
Title: Sapiens
Author: Yuval Noah Harari
Publisher: Harper Collin
Shelf: S
Please Enter 'Y'(Yes) to Confirm or 'N'(NO) Cancel:
```

```
Search book title: Sapien
('9780062316110', 'Sapiens', 'Yuval Noah Harari', 'Harper Collin', 'Avail
able', 'S')
Press ENTER to back to Search MENU
```

Use Case: Edit Book
Expect: Allow Staff and Admin to search books to edit by Title or ISBN, and edit books information, then update into database
Result: Staff and Admin can search books to edit by Title or ISBN, and edit books information, then update into database

```
(Enter '-1' to BACK)
Search book title: Sapiens
('9780062316110', 'Sapiens', 'Yuval Noah Harari', 'Harper Collin', 'Available', 'S')
Press ENTER if you want to search again
ENTER ISBN of Book you want to EDIT:
```

```
(Enter '-1' to BACK)
Search book isbn: 9780062316110
('9780062316110', 'Sapiens', 'Yuval Noah Harari', 'Harper Collin', 'Available', 'S')
Press ENTER if you want to search again
ENTER ISBN of Book you want to EDIT:
```

```
Book Information
ISBN: 9780062316110
Title: Sapiens
Author: Yuval Noah Harari
Publisher: Harper Collin
Shelf: S
(Press ENTER to continue)
```

```
Update Book Information
ISBN: 9780062316110
Title: Sapien: A Brief History of Humankind
Author: Yuval Harari
Publisher: Harper
Shelf: A
Please Enter 'Y'(Yes) to Confirm or 'N'(No) Cancel:
```

Use Case: Delete Book
Expect: Allow Staff and Admin to search books to delete by Title or ISBN, and delete books information, then update into database
Result: Allow Staff and Admin to search books to delete by Title or ISBN, and delete books information, then update into database

```
(Enter '-1' to BACK)
Search book title: kite runner
('9781594481710', 'The Kite Runner', 'Khaled Hosseini', 'Riverhead Books', 'Available
', 'J')
Press ENTER if you want to search again
ENTER isbn of Book you want to DELETE:
```

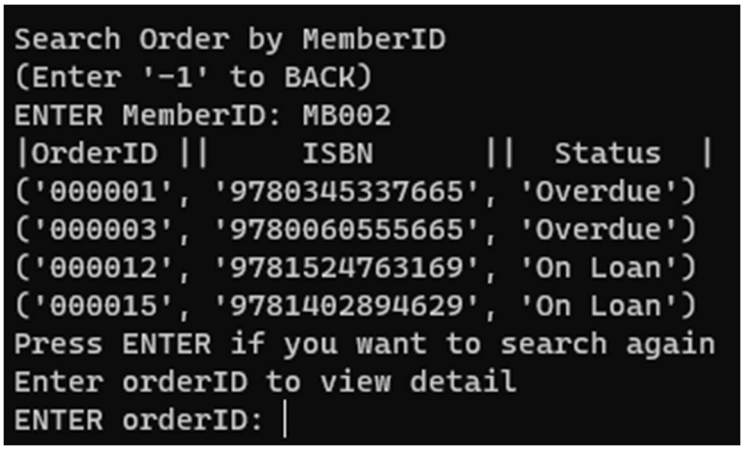
```
(Enter '-1' to BACK)
Search book isbn: 9781594481710
('9781594481710', 'The Kite Runner', 'Khaled Hosseini', 'Riverhead Books', 'Available
', 'J')
Press ENTER if you want to search again
ENTER isbn of Book you want to DELETE:
```

```
Delete Book Information
ISBN: 9781594481710
Title: The Kite Runner
Author: Khaled Hosseini
Publisher: Riverhead Books
Shelf: J
Please Enter 'Y'(Yes) to Confirm or 'N'(No) Cancel:
```

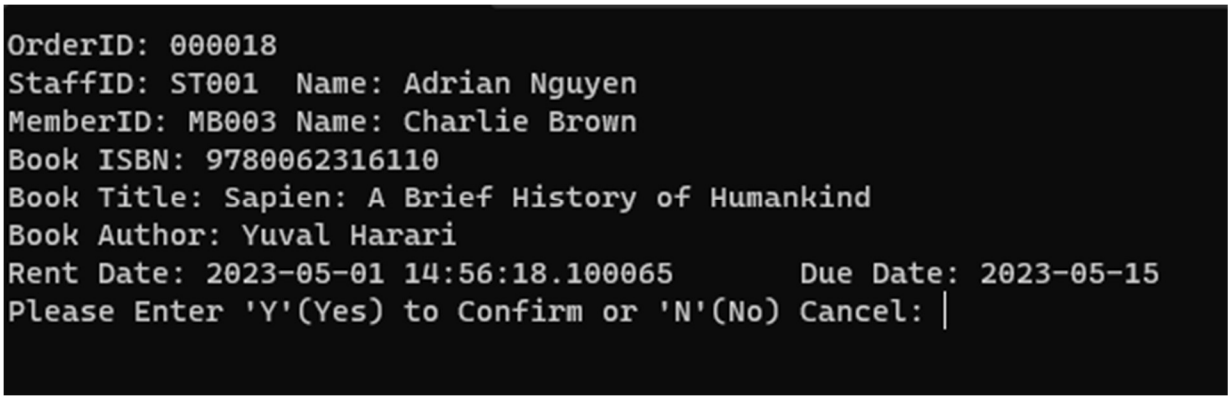
```
(Enter '-1' to BACK)
Search book isbn: 9781594481710
No books found with ISBN: 9781594481710
Press ENTER if you want to search again
ENTER isbn of Book you want to DELETE:
```

Use Case: Search Order
Expect: Allow staff and admin to search order by ISBN or MemberID, then orders' information will be displayed
Result: Orders' information is displayed after searching

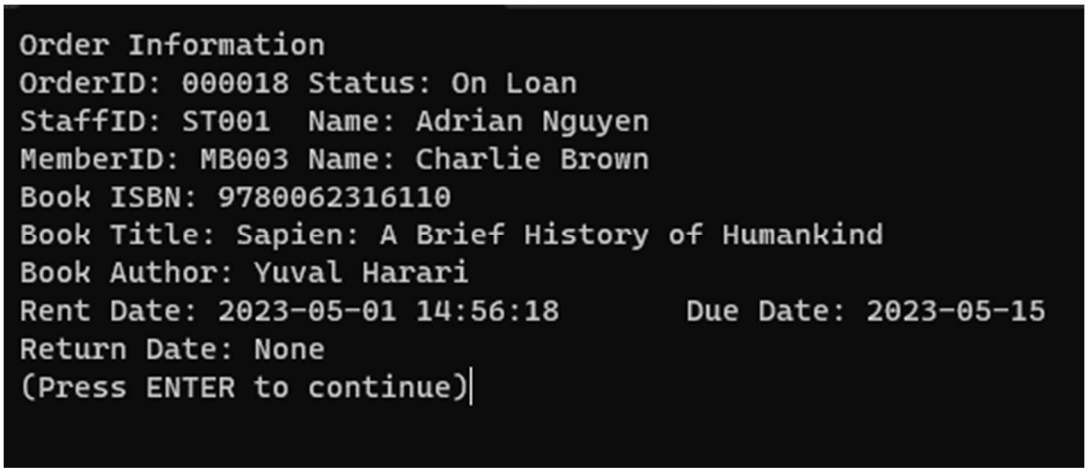
```
Search Order by ISBN
(Enter '-1' to BACK)
ENTER ISBN: 9780345337665
|OrderID ||     ISBN      ||  Status  |
('000001', '9780345337665', 'Overdue')
('000016', '9780345337665', 'On Loan')
Press ENTER if you want to search again
Enter orderID to view detail
ENTER orderID:
```

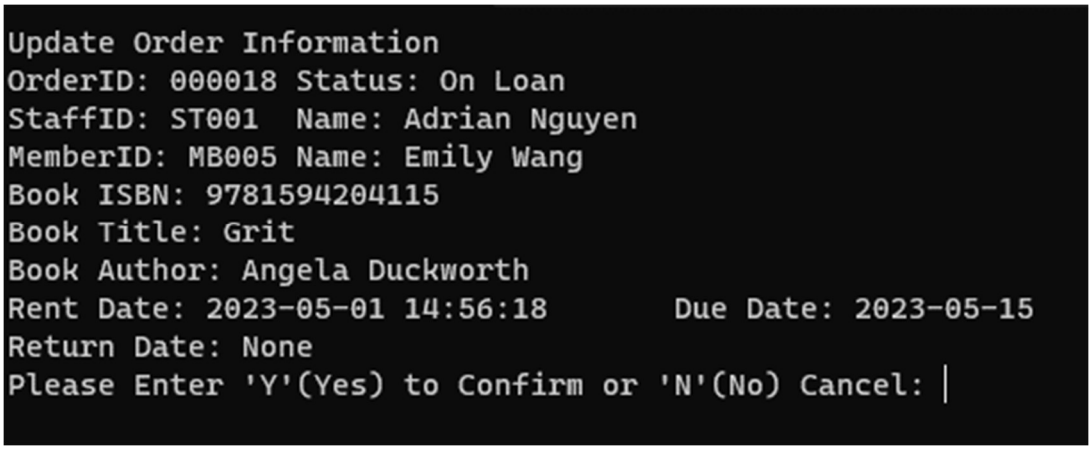


Use Case: Create New Book
Expect: Allow Staff to create new orders by input, memberID, ISBN, and the number of borrow days, then update order in the database
Result: Staff can create new orders by input, memberID, ISBN, and the number of borrow days, then update order in the database

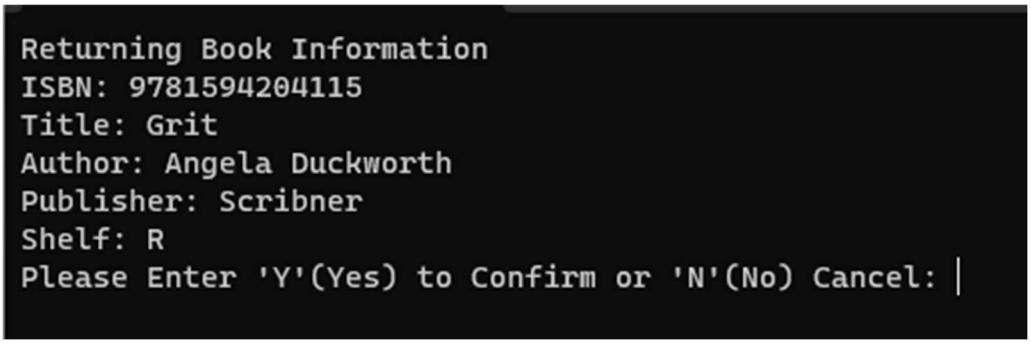


Use Case: Edit Order
Expect: Allow Staff and Admin to edit orders information, then update into database
Result: Staff and Admin can edit orders information, then update into database





Use Case: Receive Returning Book
Expect: Allow Staff and Admin update the order to indicate that it was returned, and make the book available.
Result: Staff and Admin can update the order to indicate that it was returned, and make the book available.



Use Case: Search Member
Expect: Allow Staff and Admin to search members' information by Email or Name, and members' information will be displayed

Result: members' information is displayed after being searched

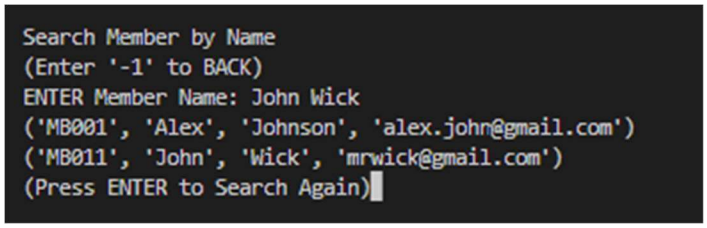
```
Search Member by Email
(Enter '-1' to BACK)
ENTER Email: alex.john@gmail.com
('MB001', 'Alex', 'Johnson', 'alex.john@gmail.com')
(Press ENTER to Search Again)
```

```
Search Member by Name
(Enter '-1' to BACK)
ENTER Member Name: Alex
('MB001', 'Alex', 'Johnson', 'alex.john@gmail.com')
(Press ENTER to Search Again)
```

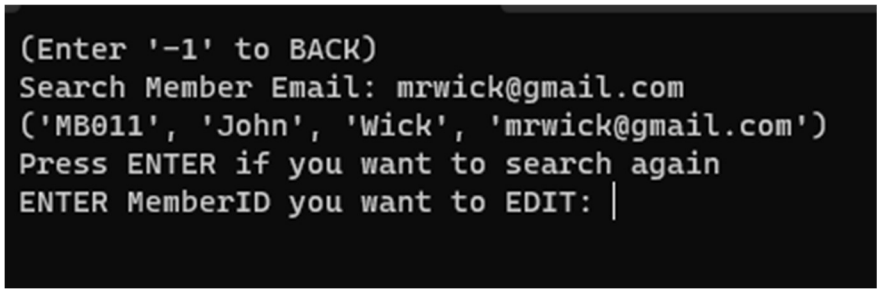
Use Case: Create New Member
Expect: Allow Staff and Admin to create new members by input name, email, password, then update to database
Result: Staff and Admin can create new members by input name, email, password, then update to database

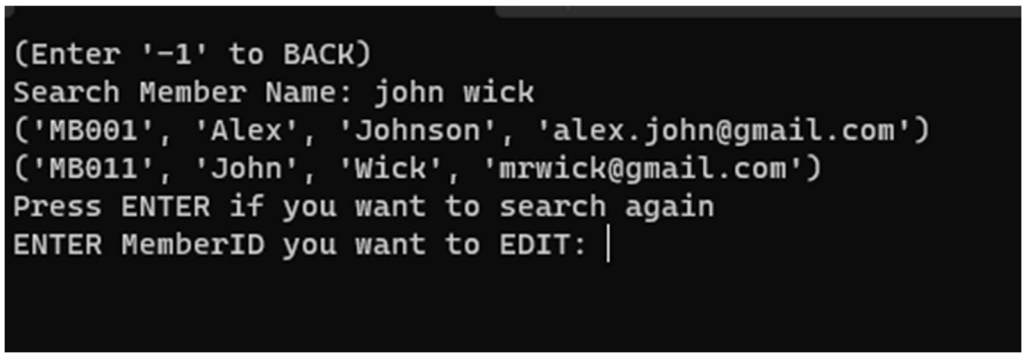
```
Please Insert New Member Information
(ENTER '-1' to Back)
Frist Name: John
Last Name: Wick
Email: mrwick@gmail.com
Password: J1234
```

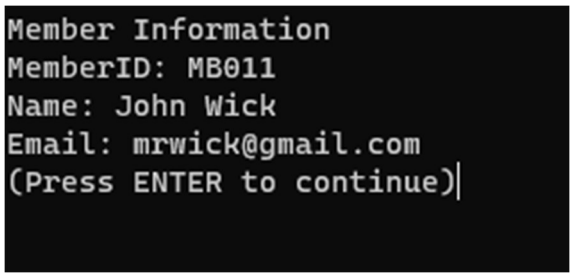
```
New Member Information
MemberID: MB011
Name: John Wick
Email: mrwick@gmail.com
Please Enter 'Y'(Yes) to Confirm or 'N'(NO) Cancel:
```



Use Case: Edit Member
Expect: Allow Staff and Admin to search for member by email or name, and edit members' information then update the database
Result: Staff and Admin can search for member by email or name, and edit members' information then update the database







```
Update Member Information
MemberID: MB011
Name: John Pooky Man Wick
Email: pookyman@gmail.com
Please Enter 'Y'(Yes) to Confirm or 'N'(No) Cancel:
```

Use Case: Delete Member
Expect: Allow Staff and Amin to search for member by email or name, and delete members, then update the database
Result: Staff and Amin can search for member by email or name, and delete members, then update to the database

```
(Enter '-1' to BACK)
Search Member Email: pookyman@gmail.com
('MB011', 'John Pooky Man', 'Wick', 'pookyman@gmail.com')
Press ENTER if you want to search again
ENTER MemberID you want to DELETE:
```

```
(Enter '-1' to BACK)
Search Member Name: wick
('MB011', 'John Pooky Man', 'Wick', 'pookyman@gmail.com')
Press ENTER if you want to search again
ENTER MemberID you want to DELETE:
```

```
Delete Member Information
MemberID: MB011
Name: John Pooky Man Wick
Email: pookyman@gmail.com
Please Enter 'Y'(Yes) to Confirm or 'N'(No) Cancel:
```

```
(Enter '-1' to BACK)
Search Member Email: pookyman@gmail.com
No Member Found
Press ENTER if you want to search again
ENTER MemberID you want to DELETE:
```

Use Case: Search Staff
Expect: Allow Admin to search staffs' information by Email or Name, and staffs' information will be displayed
Result: members' information is displayed after being searched

```
Search Staff by Email
(Enter '-1' to BACK)
ENTER Email: CTran@gmail.com
('ST004', 'Cooper', 'Tran', 'CTran@gmail.com')
(Press ENTER to Search Again)
```

```
Search Staff by Name
(Enter '-1' to BACK)
ENTER Staff Name: Cooper Tran
('ST004', 'Cooper', 'Tran', 'CTran@gmail.com')
(Press ENTER to Search Again)
```

Use Case: Create New Staff
Expect: Allow Admin to create new staffs by input name, email, password, then update to database
Result: Admin can create new staffs by input name, email, password, then update to database

```
New Staff Information
StaffID: ST005
Name: Peter Parker
Email: PPspider@gmail.com
Please Enter 'Y'(Yes) to Confirm or 'N'(NO) Cancel:
```

Use Case: Edit Staff
Expect: Allow Admin to search for staffs by email or name, and edit staffs's information then update the database
Result: Admin can search for staffs by email or name, and edit staffs's information then update the database

```
(Enter '-1' to BACK)
Search Staff Email: ppspider@gmail.com
('ST005', 'Peter', 'Parker', 'PPspider@gmail.com')
Press ENTER if you want to search again
ENTER StaffID you want to EDIT:
```

```
(Enter '-1' to BACK)
Search Staff Name: Peter Parker
('ST005', 'Peter', 'Parker', 'PPspider@gmail.com')
Press ENTER if you want to search again
ENTER StaffID you want to EDIT:
```

```
Staff Information
StaffID: ST005
Name: Peter Parker
Email: PPspider@gmail.com
(Press ENTER to continue)
```

```
Update Staff Information
StaffID: ST005
Name: Peter Spidy Parker
Email: spidyparker@gmail.com
Please Enter 'Y'(Yes) to Confirm or 'N'(No) Cancel:
```

Use Case: Delete Staff
Expect: Allow Amin to search for staffs by email or name, and delete staffs, then update the database
Result: Staff and Amin can search for staffs by email or name, and delete staffs, then update the database

```
(Enter '-1' to BACK)
Search Staff Email: spidyparker@gmail.com
('ST005', 'Peter Spidy', 'Parker', 'spidyparker@gmail.com')
Press ENTER if you want to search again
ENTER StaffID you want to DELETE:
```

```
(Enter '-1' to BACK)
Search Staff Name: Peter Spidy
('ST005', 'Peter Spidy', 'Parker', 'spidyparker@gmail.com')
Press ENTER if you want to search again
ENTER StaffID you want to DELETE:
```

```
Delete Staff Information
StaffID: ST005
Name: Peter Spidy Parker
Email: spidyparker@gmail.com
Please Enter 'Y'(Yes) to Confirm or 'N'(No) Cancel:
```

```
(Enter '-1' to BACK)
Search Staff Name: Peter Spidy Parker
No Staff Found
Press ENTER if you want to search again
ENTER StaffID you want to DELETE:
```

Use Case: Generate Report
Expect: Allow Admin to generate a report and displayed the report
Result: The report is generated and displayed

```
****CodeX Library Report****
Time: 2023-05-01 16:05:39.240609
Total Amount of Availabile Books: 14
Total Amount of On Loan Books: 17
Total Amount of Current On Loan Order: 10
Total Amount of Current Overdue Order: 7
Total Current Member: 10
Total Current Staff: 4
****Staff Order****
ID: ST001 Name: Adrian Nguyen ||| Total Order: 2
ID: ST002 Name: Cerian Jaime ||| Total Order: 5
ID: ST003 Name: Nathan Nguyen ||| Total Order: 0
ID: ST004 Name: Cooper Tran ||| Total Order: 3
(Press ENTER to Back)
```

X. User Documentation

Welcome to our Library Management System!

Our system provides a comprehensive platform for admins, staff, and members to manage their accounts and access different features based on their user roles. Whether you're looking to manage books, orders, members, or generate reports, our system has got you covered. Before you get started here is some helpful information to get started.

System Requirements:

- Operating System: Windows, macOS, or Linux
- Python 3.x or higher installed
- MySQL database management system installed.
- Enough storage to store the database and program files.

Installation Instructions:

1. Download the latest version of Python from the official website and install it on your computer.
2. Download and install the MySQL database management system on your computer.
3. Download the source code for the library management system from our GitHub repository.
4. Extract the source code files from the downloaded archive to a directory on your computer.
5. Create a new MySQL database and import the database schema provided with the source code.
6. Edit the configuration file of the library management system to configure the database connection settings.
7. Open a terminal or command prompt and navigate to the directory where the source code files are located.
8. Run the main Python file of the library management system using the following command: ***python library.py.***

Getting Started/Main Menu:

1. Open the program and you will be prompted with a main menu.

2. Select your user role - admin, staff, or member.
3. Enter your email and password to login.

Once logged in, you will be taken to a menu specific to your user role, you will now be able to roam around our Library System freely.

Admin Menu:

1. Manage Book - Allows you to add, edit, and delete books from the library.
2. Manage Order - Allows you to view and manage orders placed by members.
3. Manage Member - Allows you to view and manage member accounts.
4. Manage Staff - Allows you to view and manage staff accounts.
5. Generate Report - Allows you to generate a report on the library's book and member data.
6. Edit Personal Information - Allows you to edit your personal information, such as your name and password.

Staff Menu:

1. Manage Book - Allows you to add, edit, and delete books from the library.
2. Manage Order - Allows you to view and manage orders placed by members.
3. Manage Member - Allows you to view and manage member accounts.
4. Edit Personal Information - Allows you to edit your personal information, such as your name and password.

Member Menu:

1. Search Book - Allows you to search for books in the library and view their details.
2. View Your Order - Allows you to view your current and past orders.
3. Edit Personal Information - Allows you to edit your personal information, such as your name and password.

To logout, enter '-1' in the menu.