CMSC 495 6380 PROJECT DESIGN

GROUP 7

Franklin Pokam

Ayodeji Onitilo

Arnaud Tako

REVISION HISTORY TABLE

DATE	NAME	DESCRIPTION
06/12/2022	Ayodeji Onitilo	Subsystems, event trace diagram, scenarios
06/13/2022	Franklin Pokam	Class design, Background, Sequence diagram for each scenario, pre-condition, post-condition, and description for each sequence
06/13/2022	Arnaud Tako	Unresolved risk and mitigation
06/25/2022	Ayodeji Onitilo	Startup, shutdown and error handling scenarios, Event trace diagram
07/03/2022	Ayodeji Onitilo	Pre-condition and post-condition for each scenario, sequence diagrams and descriptions

Background

The students from group 4 in CMSC 495 at the University of Maryland Global Campus are collaborating to build a library management system. It will be a very simple application with students having the ability to borrow and return previously borrowed books. Administrators, including librarians will be able to issue books, register new students, issue books to students, check users' accounts, and update books and users in the system. This project design will focus on creating subsystem diagrams and context diagrams.

Sequence Diagrams For All Subsystems and Classes

1- Register new user

Description:

- This can be performed by users to register in order to gain access to the library.

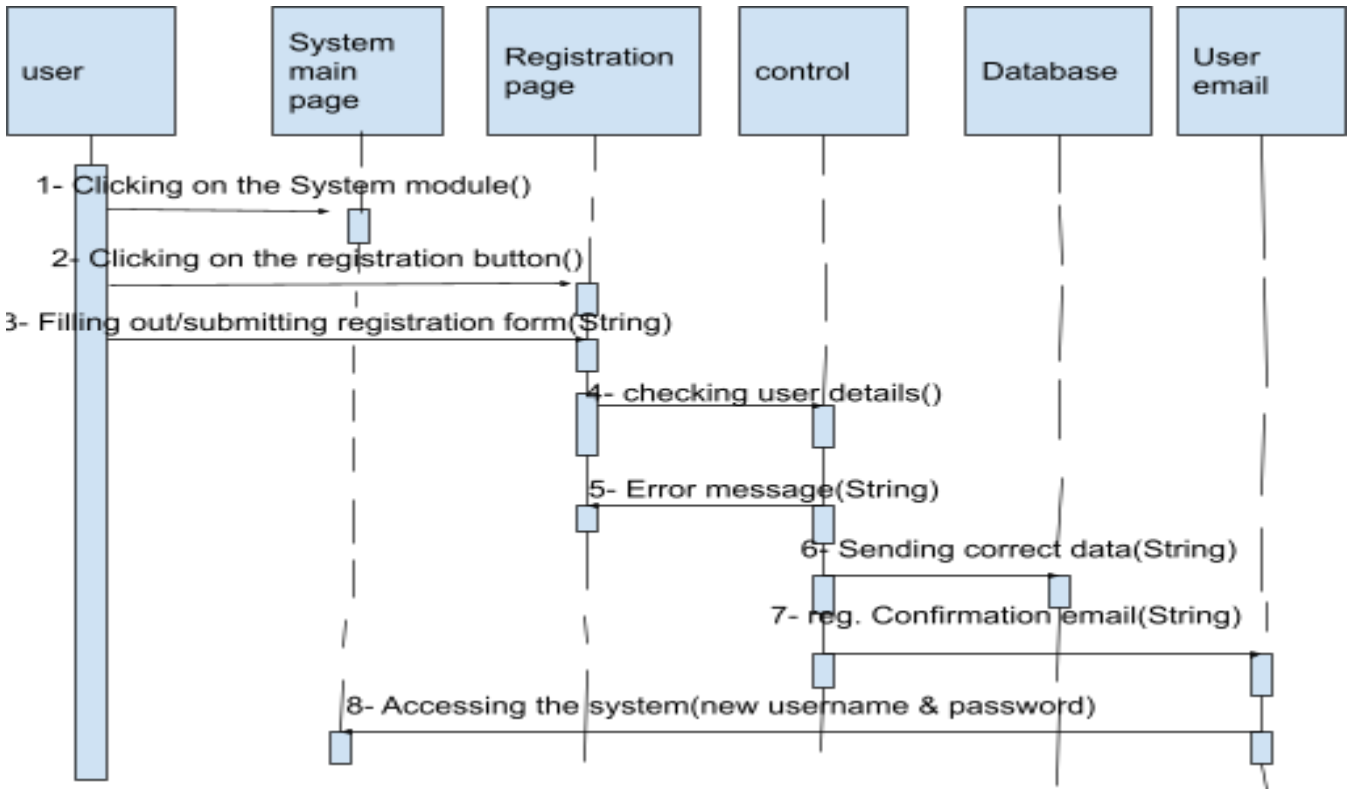
Pre-condition:

- The user shall not have a preexisting account.
- The register user button shall be available.
- The user shall have an email address and an ID number.

Post condition:

- System shall verify information.
- System shall delete wrong information.
- The username shall be unique.

Sequence diagram:



2- User login

Description:

- This is used by the user to login into the system Username and passwords are required. Those sensitive information are then verified and validated by the system. Users with invalid credentials are not allowed to login into the system.

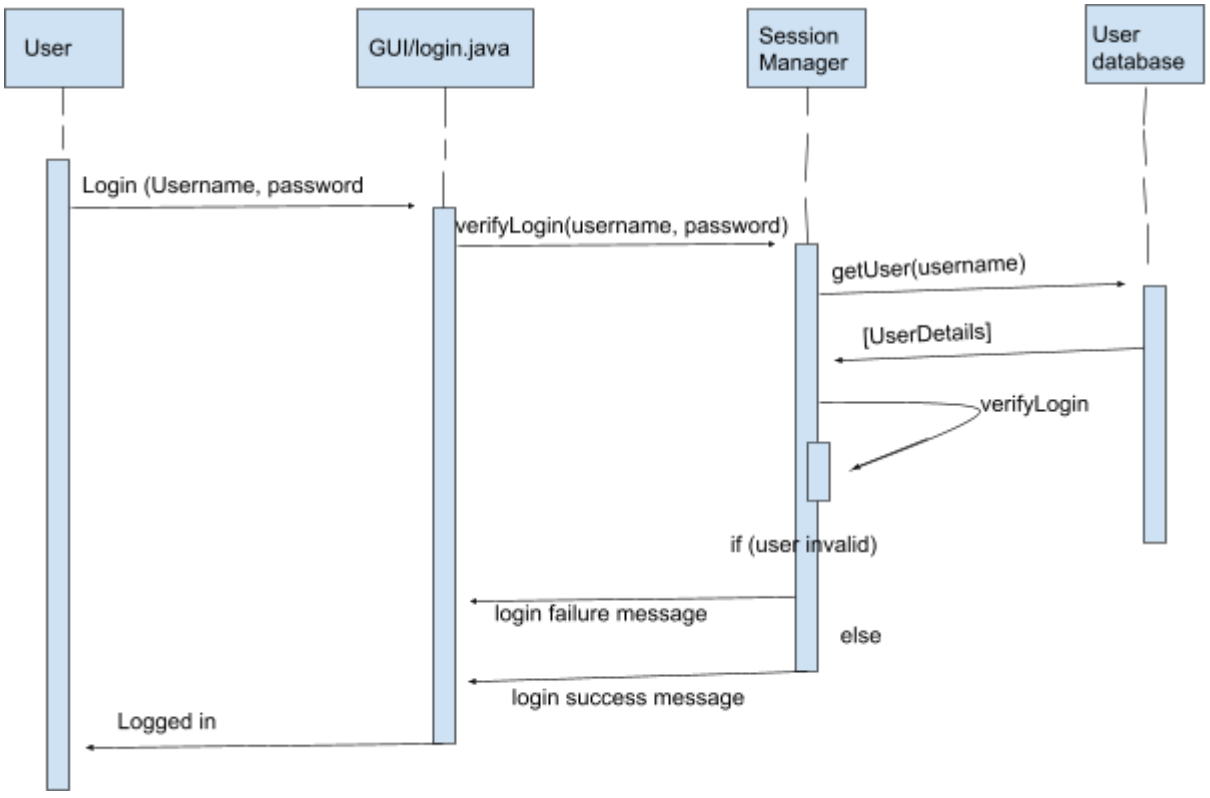
Pre-condition:

- The user shall be registered to login.
- The user shall have valid user's credentials.
- The user shall use the login page to login.

Post-condition:

- Users shall be able to log out after using the system
- Only users with valid credentials shall be allowed to login into the system. The entered credentials must match the already existing credentials in the system.
- The system shall perform an authorization verification that decides whether the user is an administrator, or a student for accessibility purposes.

Sequence Diagram:



3 - Issue book

Description:

- This functionality allows the librarian to issue books to users. It also allows the librarian to view the list of books that have been issued, and to update the student's account, and the book database.
- The librarian checks the availability of the book, and validates the student from the student record. If found the student is validated and the librarian can issue the book to him/her.

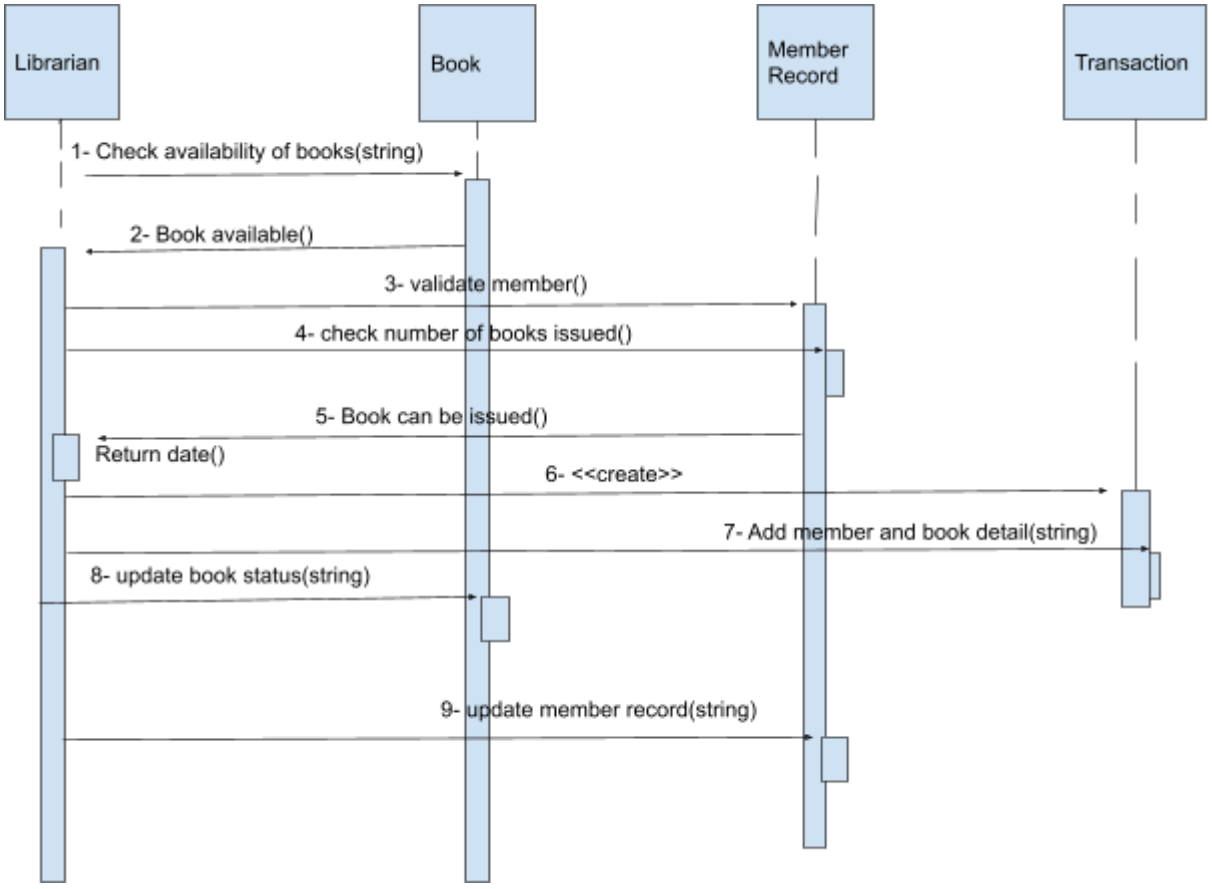
Pre-condition:

- The user shall be logged in.
- Only available books can be checked out.
- The user shall verify the availability of the book to be checked out.
- The user shall have the appropriate privilege.

Post-condition:

- The system shall provide return date information.
- The system shall verify the availability of books before issuance.
- The system shall update the number of books still available after issuance.
- The system shall be able to access the database.
- The system shall provide an error message if the book is not available.

Sequence Diagram



4- Return books

Description:

- The system allows the librarian to accept returned books from students. It also allows the librarian to view the list of books that have been issued to the students, and to update the student's account., and the book database.
- The validate validates the student from the student record, and then searches for the member details. If valid, the book is returned. A fine is added if late, and the book and student status are updated.

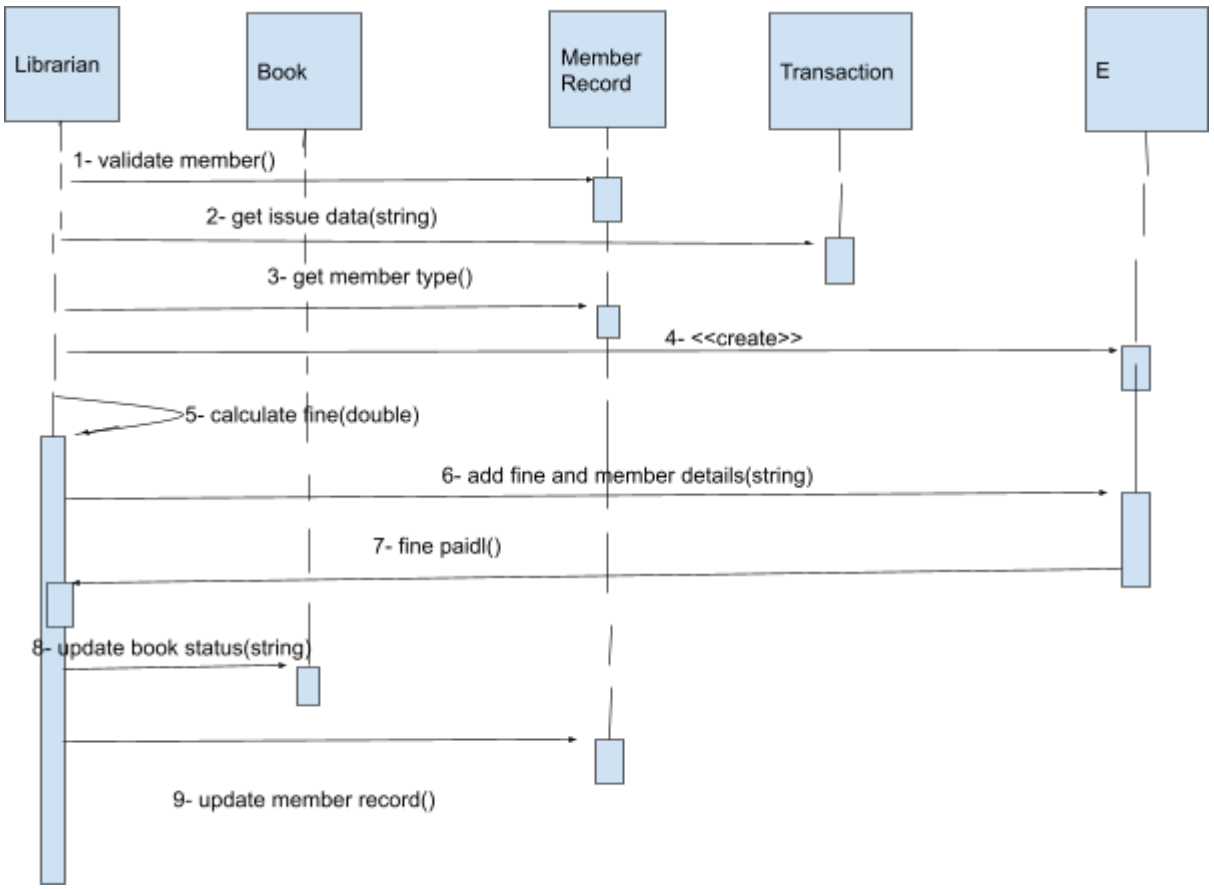
Pre-condition:

- The student shall return the physical book to the librarian.
- The user shall be logged in.
- The user shall use the book return button.
- The book details shall be in the database. It must be a book that belongs to the library.
- The user shall have the appropriate privilege.

Functional Requirements:

- The system shall provide return date information.
- The system shall verify the existence of books before acceptance.
- The system shall update the number of books available.
- The system shall be able to access the database.

Sequence Diagram:



5- Search books

Description:

- This function allows administrators and students to search books using the search method and string variables such as: isbn numbers, book titles, publication year, author's name and more.

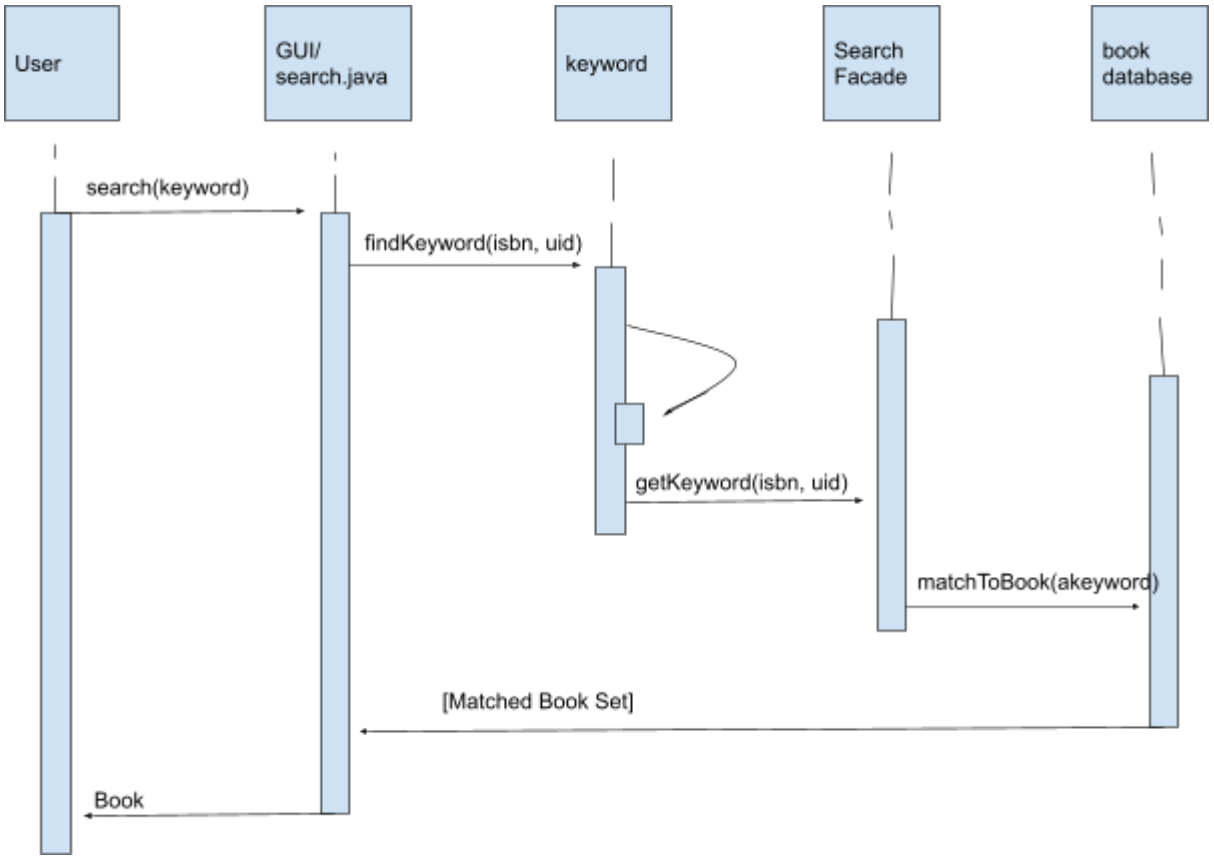
Pre-condition:

- The user shall be logged in.
- The search button shall be available to search books.
- The user must have the appropriate privilege to search books.
- There shall be a database with all the books that can be searched.

Post-condition:

- The system shall filter books based on search/keyboard entry.
- The system shall display the filtered books in a table.
- The system shall search the database using the search entry from the keyboard.
- The system shall return an error message if the search key isn't a match.

Sequence Diagram



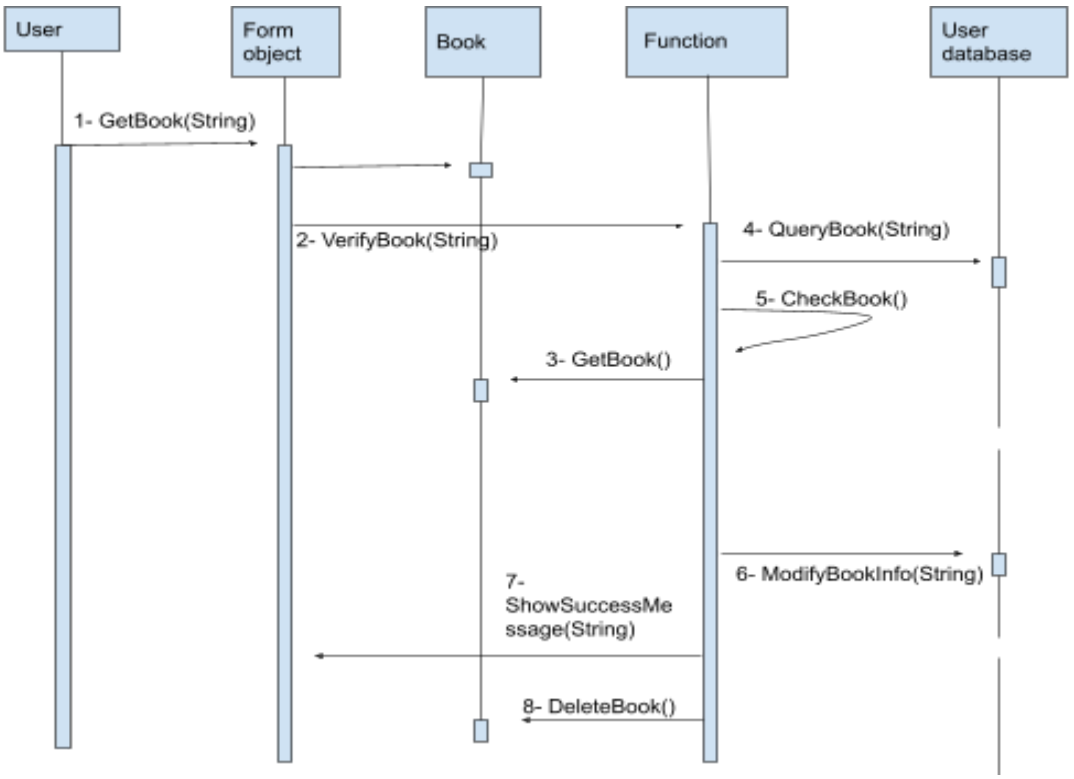
6- Renew books

Description:
This function is there to allow a user to renew a book that was already issued.

Pre-condition:
- The book shall be issued.
- The user shall be logged in.
- The user shall have the necessary privilege to complete the task.
- The book must be from the library.
- The student must be registered.

Post-condition:
- A message appears on the screen to confirm the renewal.
- The user shall update accounts.

Sequence Diagram:



7- Fines: late fees.

Description:

- This functionality allows the librarian to add a fine to a student's account if the student fails to return the book by the due date provided at the time of issuance. This functionality also allows the librarian to check the student's record.

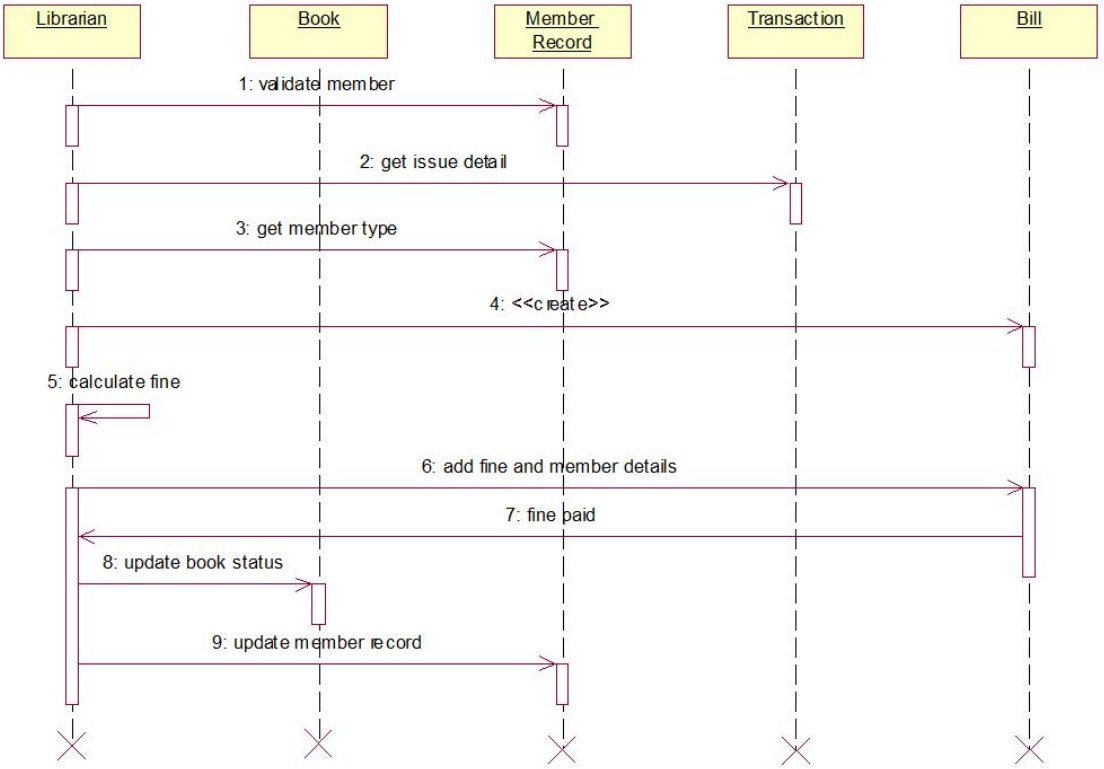
Pre-condition:

- The student shall not have returned the book by the return date.
- The user must have the appropriate privilege.
- A students' database and book shall be available.
- An add fine button shall be available.

Post-condition:

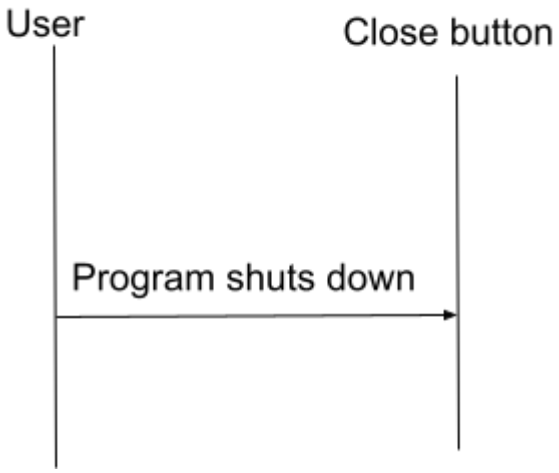
- The user shall be able to update the student's record.
- A message shall notify the student\ of the penalty.
- A success message shall be displayed on the screen.

Sequence Diagram:



Shutdown scenario

The user has an exit button where they can close out of the library management screen when they're done

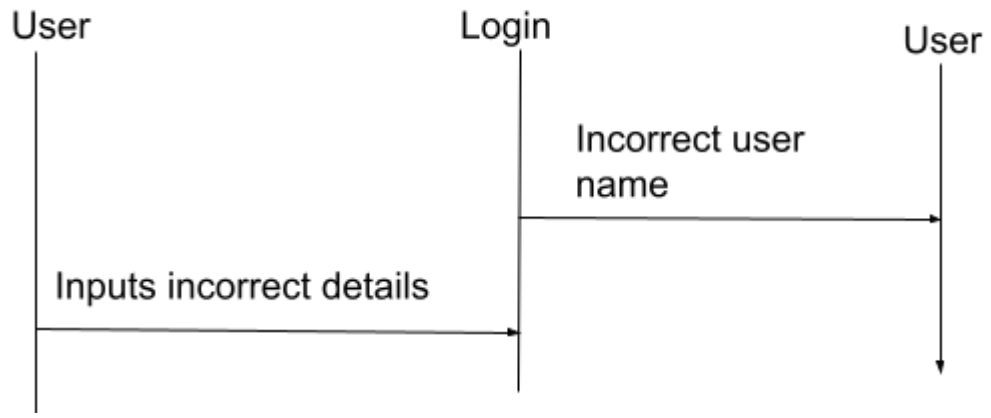


- Pre-condition - The user clicks the exit button to close out of the program
- Post-condition - The running program will now be closed

Error handling scenario

When a user inputs invalid values when searching for a book, the program throws an exception

- Pre-condition - The user inputs an invalid value when trying to search for a book
- Post-condition - The program can display an error message to the user



Class Design:

```
class BookDetails {
        private String id;
        private String authorName;
        private publicationDate;

        public BookDetails {
```

```
                this.id = id;
                this.authorName = authorName;
                this.publicationDate = publicationDate;
        }

        Call id.getId();

        Call authorName.getAuthorName();

        Call publicationDate.getPublicationDate();


}
.....

class Book{
        public Book(String bookItemId, BookDetails bookDetails) {
                this.bookId = bookId;
                this.bookDetails = bookDetails;
        }

        private String bookId;

        private String bookDetails;

        Call bookId.getBookId();


        Call bookDetails.getBookDetails();

}
.....

public enum UserType {
        LIBRARIAN, MEMBER
}
```

```
.....
class User {
	public User(String barcode, String name, UserType userType) {
		this.barcode = barcode;
		this.name = name;
		this.userType = userType;
	}

	private String barcode;
	private String name;
	private UserType userType;`

	Call barcode.getBarcode();


	Call userType.getUserType();


	Call name.getName();

}
.....
class Main or Library {

	private User currentUser;
	private UserRepository userRepository;
	private BookCatalog bookCatalog;
	private LendingService lendingService;

	private void ensureLoggedInUser() {

	}

	private void ensureLibrarianAccess() {
		ensureLoggedInUser();
		if (currentUser.getUserType() != UserType.LIBRARIAN) {
			throw new RestrictedAccessException();
```

```
        }
}

public void loginUser(String) {

        this.currentUser = user;
}

public void createUser(User newUser) {
        ensureLibrarianAccess();

        userRepository.addUser(newUser);
}

public void removeUser(String) {
        ensureLibrarianAccess();

        userRepository.removeUser();
}

public void addBook() {
        ensureLibrarianAccess();

        bookCatalog.addBook(book);
}

public void removeBook(String bookID) {
        ensureLibrarianAccess();

        bookCatalog.removeBook(bookID);
}

public List<BookDetails> searchByTitle(String title) {
        ensureLoggedInUser();

        return bookCatalog.searchByTitle(title);
}
```

```
public List<BookDetails> searchByAuthor(String name) {
        ensureLoggedInUser();

        return bookCatalog.searchByAuthor(name);
}

public List<BookDetails> searchByPublicationDate(publicationDate) {
        ensureLoggedInUser();

        return bookCatalog.searchByPublicationDate(publicationDate);
}

public List<BookItem> getCheckoutBooks() {
        ensureLoggedInUser();

        return lendingService.getCheckedOutBooks(currentUser);
}

public List<User> getLendingUsers(String bookID){
        ensureLibrarianAccess();

        return lendingService.getLendingUsers(bookID);
}

public int getOverdueFines() {
        ensureLoggedInUser();
        return lendingService.getOverdueFineAmount(currentUser);
}

public BookItem checkout(String bookID) {
        ensureLoggedInUser();

        return lendingService.checkout(currentUser, bookID);
}

public boolean renew(Book) {
```

```
        ensureLoggedInUser();

        return lendingService.renew(currentUser);
    }

    public boolean returnBookItem(Book) {
        ensureLoggedInUser();

        return lendingService.returnBook(currentUser);
    }
```

POSSIBLE ENHANCEMENTS AND RISK MITIGATION

The risk management plan starts with collecting data about the system and users who will use the system. This helps in understanding the potential scenarios in the future. The analysis of the time of system access, size of data stored in the system, sensitivity of data, number of users in a day, etc. can offer insights pertaining to plan development. When it comes to a library management system, a fundamental approach should be developed for risk management. It is also a good idea to have a risk manager who can plan regular updates and check up regularly on certain things.

- The budgeting is a very important factor here. If fines are involved, the budget should be discussed for risk management.
- The system vulnerabilities should be checked regularly. The assets related to the library system should be identified.
- Other types of threats include failure of hardware, software, wiring, etc. Data corruption should be avoided by backing up data on some external storage. Firewalls and antivirus software should be used to avoid any malicious intervention.
- The program can be enhanced by having security measures in place, such as 2FA or MFA to prevent an unauthorised user from accessing a user's account.
- User accounts can possibly have a timeout session after a period of inactivity to prevent an unauthorised user from accessing a user's account