# **SLTC re:Start Bootcamp**

# **Final Assignment (Group project)**

# Project name: Library Management system for a small school

# Team members:

|  |  |
| --- | --- |
| **Student ID** | **Full Name** |
| RS19113 | M S M Shazir |
| RS19123 | W Iresha U Silva |
| RS19122 | F.M. N Aysha |

# GitHub URL: <https://github.com/BootcampProgram/NewLibraryManagementSystem>

# Demo URL (if available):

Contents

[**SLTC re:Start Bootcamp** 1](#_Toc27656719)

[**Final Assignment (Group project)** 1](#_Toc27656720)

[Project name: Library Management system for a small school 1](#_Toc27656721)

[Team members: 1](#_Toc27656722)

[GitHub URL: https://github.com/BootcampProgram/NewLibraryManagementSystem 1](#_Toc27656723)

[Demo URL (if available): 1](#_Toc27656724)

[Technical documentation 3](#_Toc27656725)

[Introduction 3](#_Toc27656726)

[Use case diagram 4](#_Toc27656727)

[Navigation map 5](#_Toc27656728)

[Sample UI 7](#_Toc27656729)

[Database relationship (ER) diagram 15](#_Toc27656730)

[Business logic 16](#_Toc27656731)

[Backend End Points 17](#_Toc27656732)

[Final screenshots 24](#_Toc27656733)

# Technical documentation

## Introduction

The project Library Management System aims at developing a fully functional computerized system to maintain all the day to day activity of a library. The main aim of this project is providing an easy to handle and automated Library Management System. Manual Library Management System involves lots of paperwork as the Library information including user data, librarian data, book records, among others are all documented manually. Also issuing and returning books are done manually leading to waste of time and resources from the users and library authority.

This project has many features which are generally not available in normal Library Management Systems like facility of user login and facility of librarian login. It also has a facility of admin login through which the admin can monitor the whole system. Books and students maintenance modules are also included in this system which would keep track of the students using the library and also a detailed description about the books a library contains. With this computerized system there will be no loss of book record or member record which generally happens when a non-computerized system is used.

This project also provides features and interfaces for maintaining librarian’s record, student’s record, student’s history of issue and return, reservations etc. The librarian can create, view, update and delete book details. And also he can issue the books to the students and maintain their records. It shows trending book among the students too. Searching, adding and modifying information has been made easy, so the librarian doesn’t have to be an expert in Library administration.

In this project we can maintain the late fine of students who returns the issued books after the due date. This system allows the students to view the book details, reserve books, and add books to wish list. The goal of this project is to provide simplicity as well as security and efficiency to the management. This system has user friendly interface, fast access to database, less errors, more storage capacity, search facilities as advantages.

Overall this project is being developed to help the students as well as staff of library to maintain the library in a best way and to reduce the human effort.

## Use case diagram

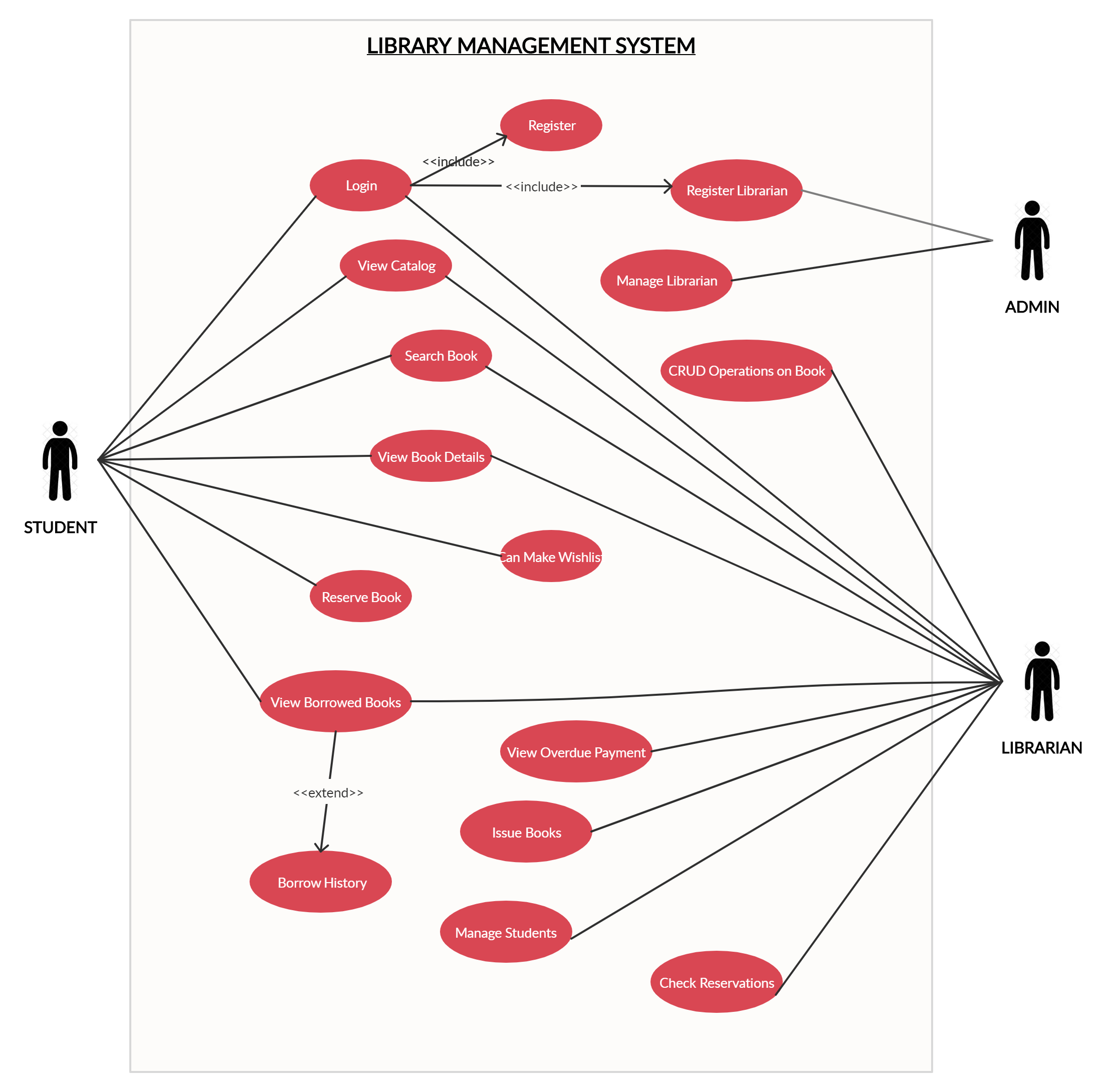


Figure 1: Use case diagram

## Navigation map

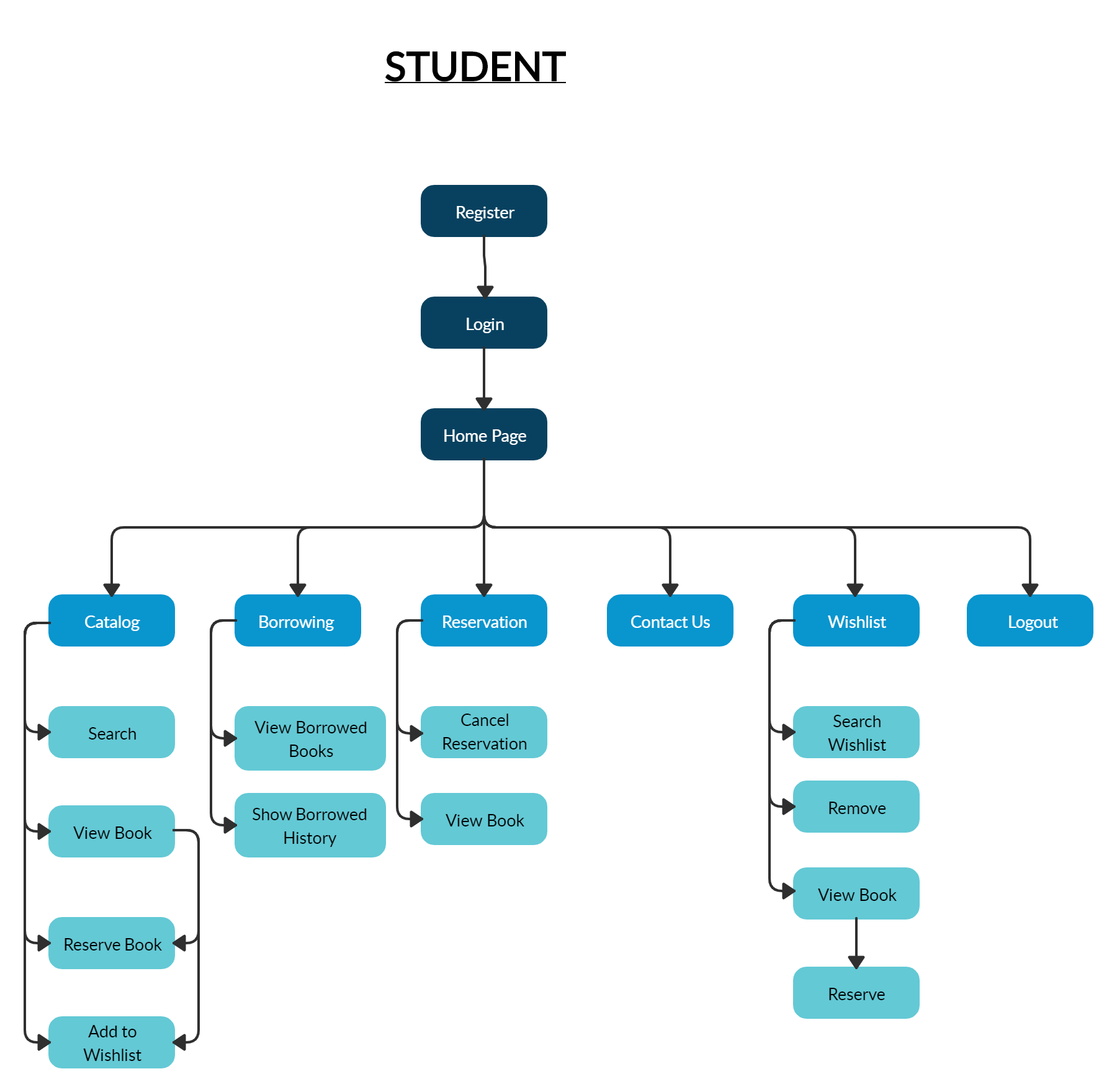


Figure 2: Navigation map of a student

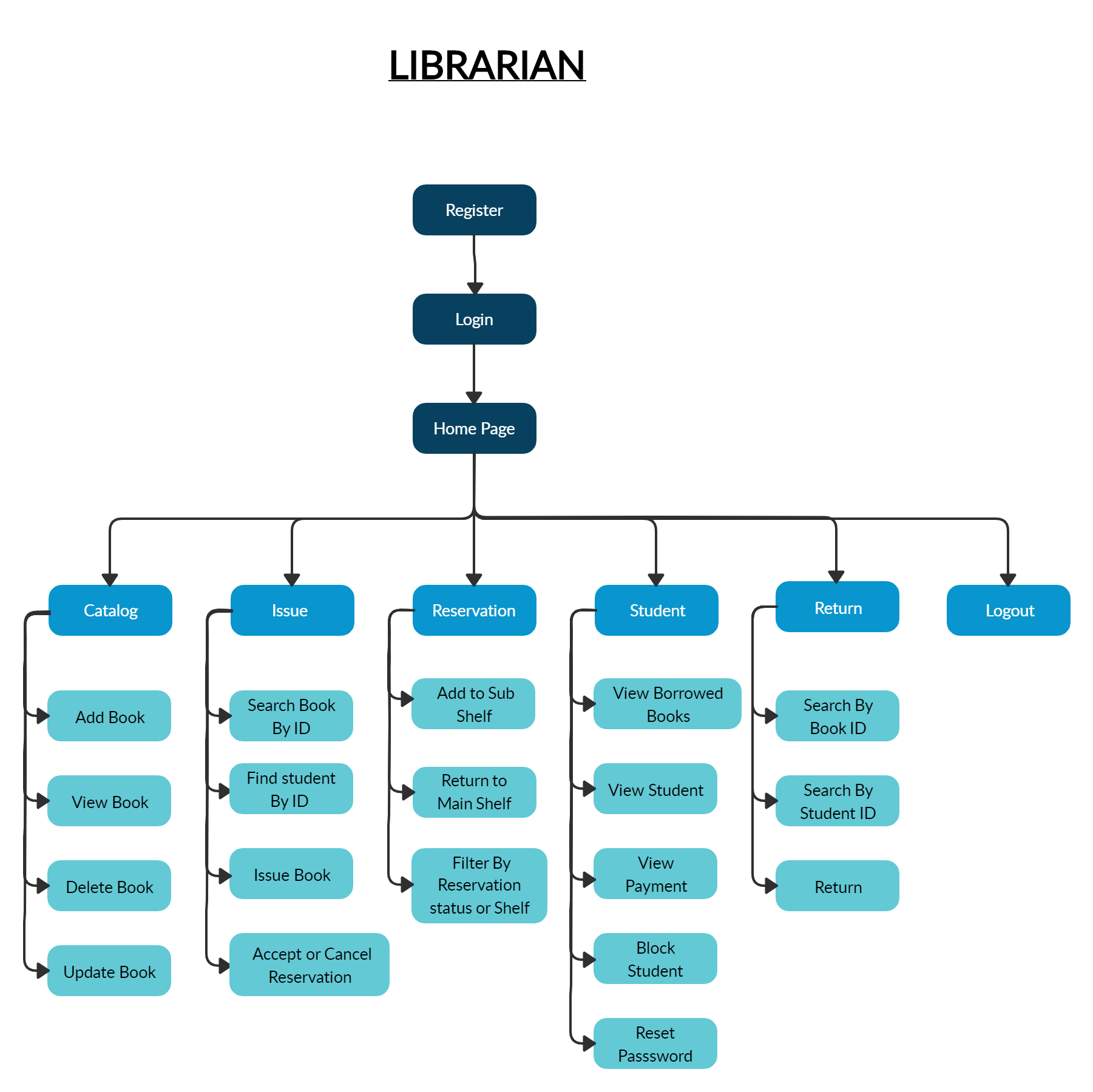


Figure 3: Navigation map of a librarian

## Sample UI

**Student’s Sample UI**

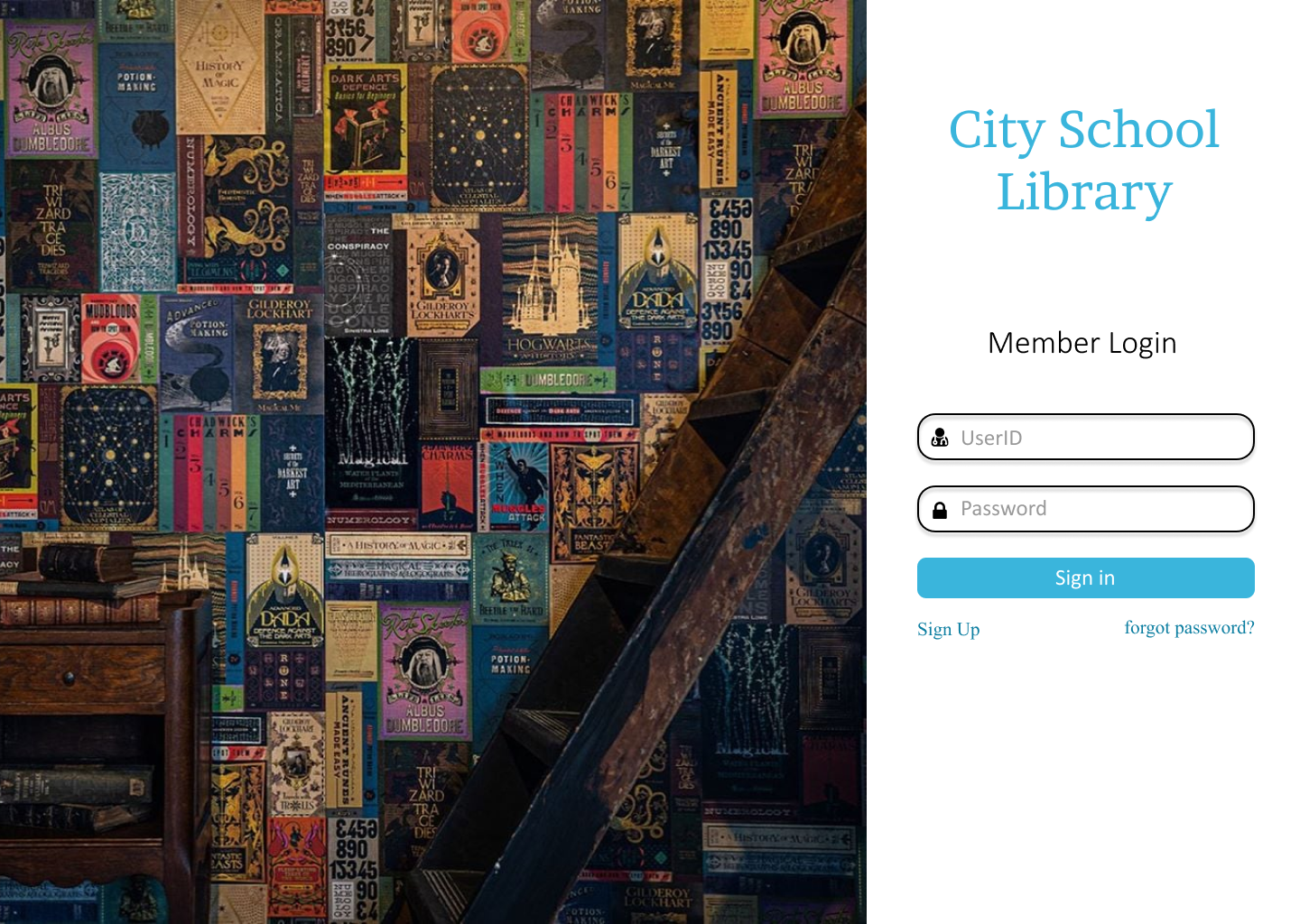


Figure 4: Login page

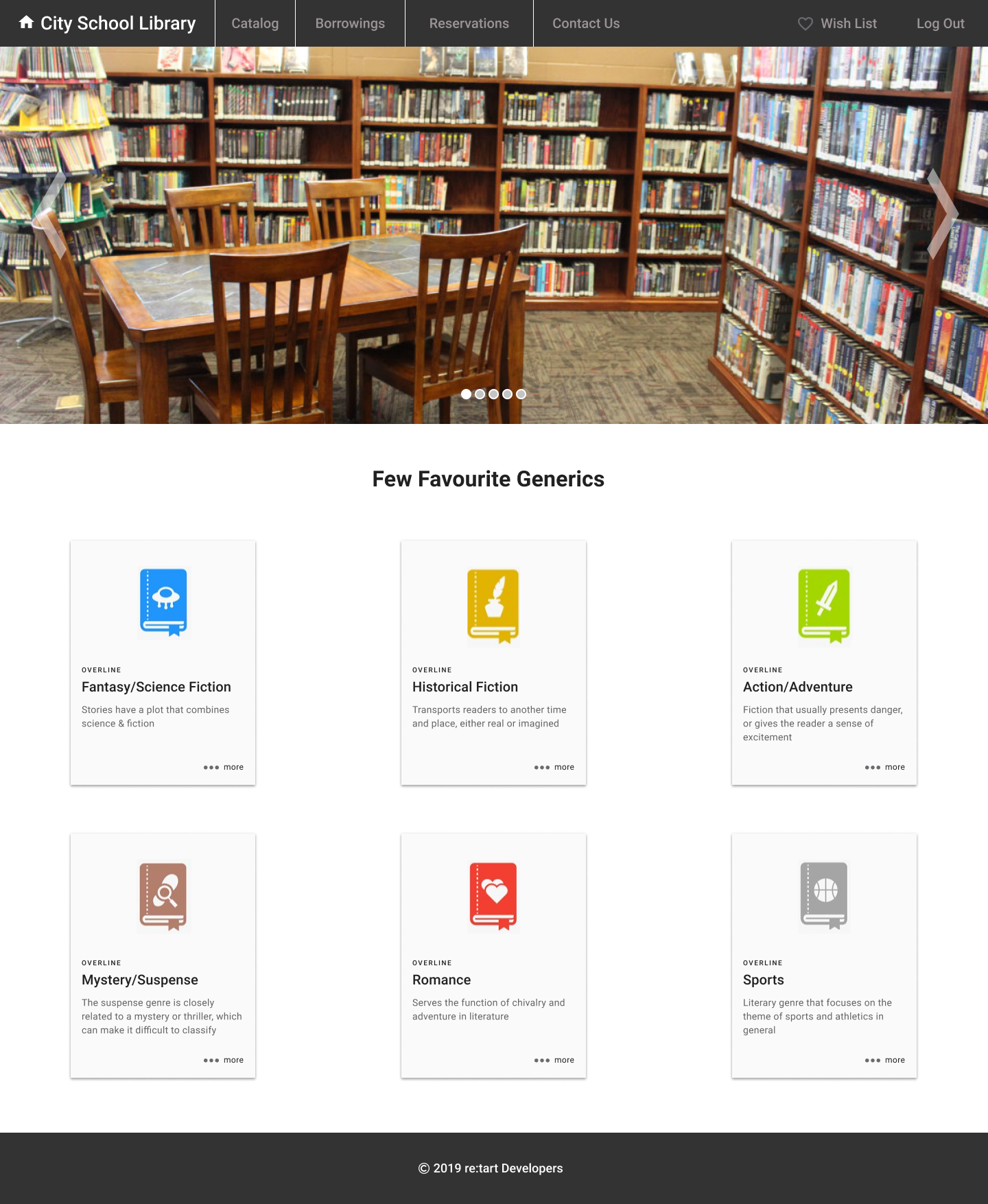


Figure 5: Home page of student UI

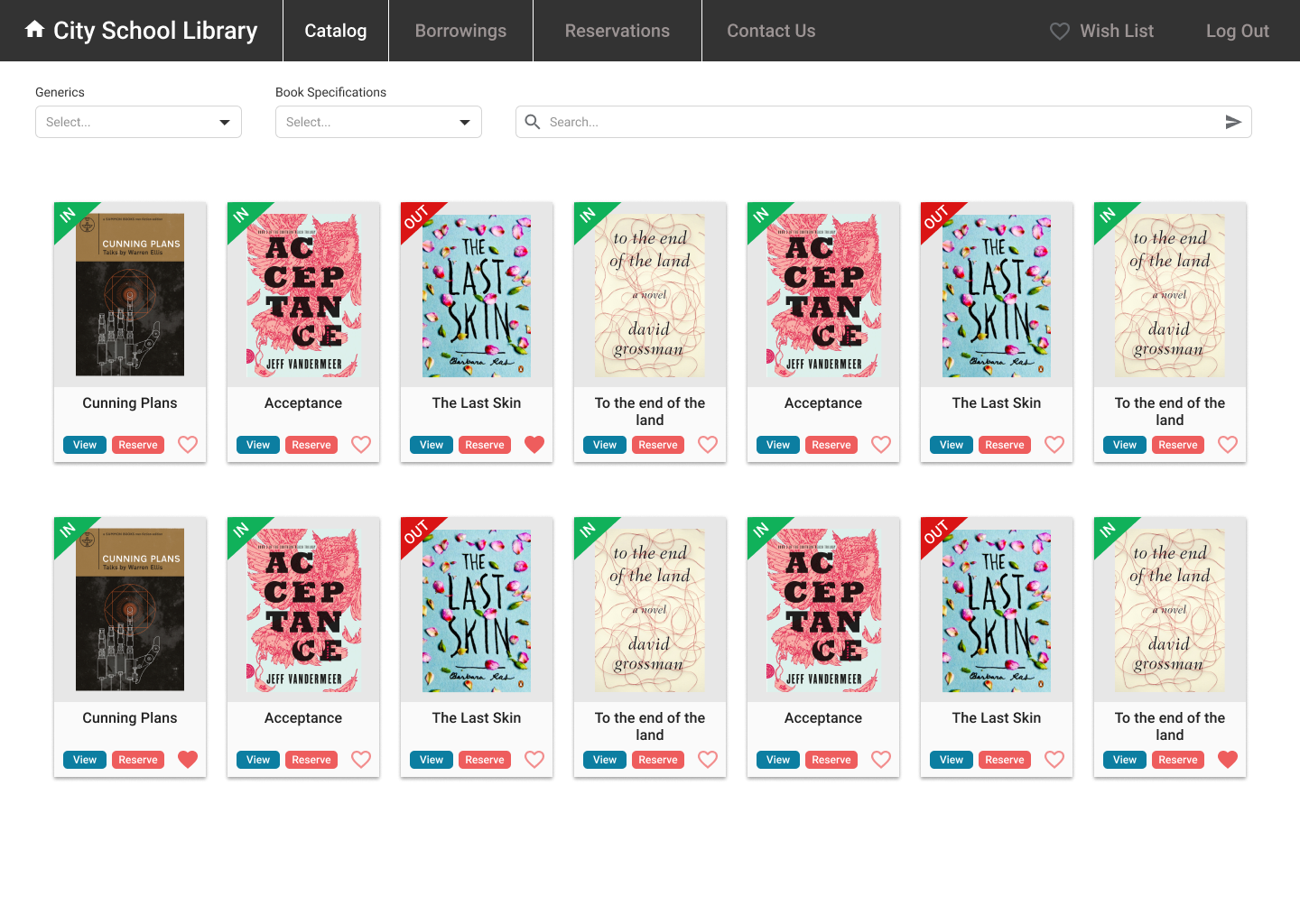


Figure 6: Library catalog page for view and search all books

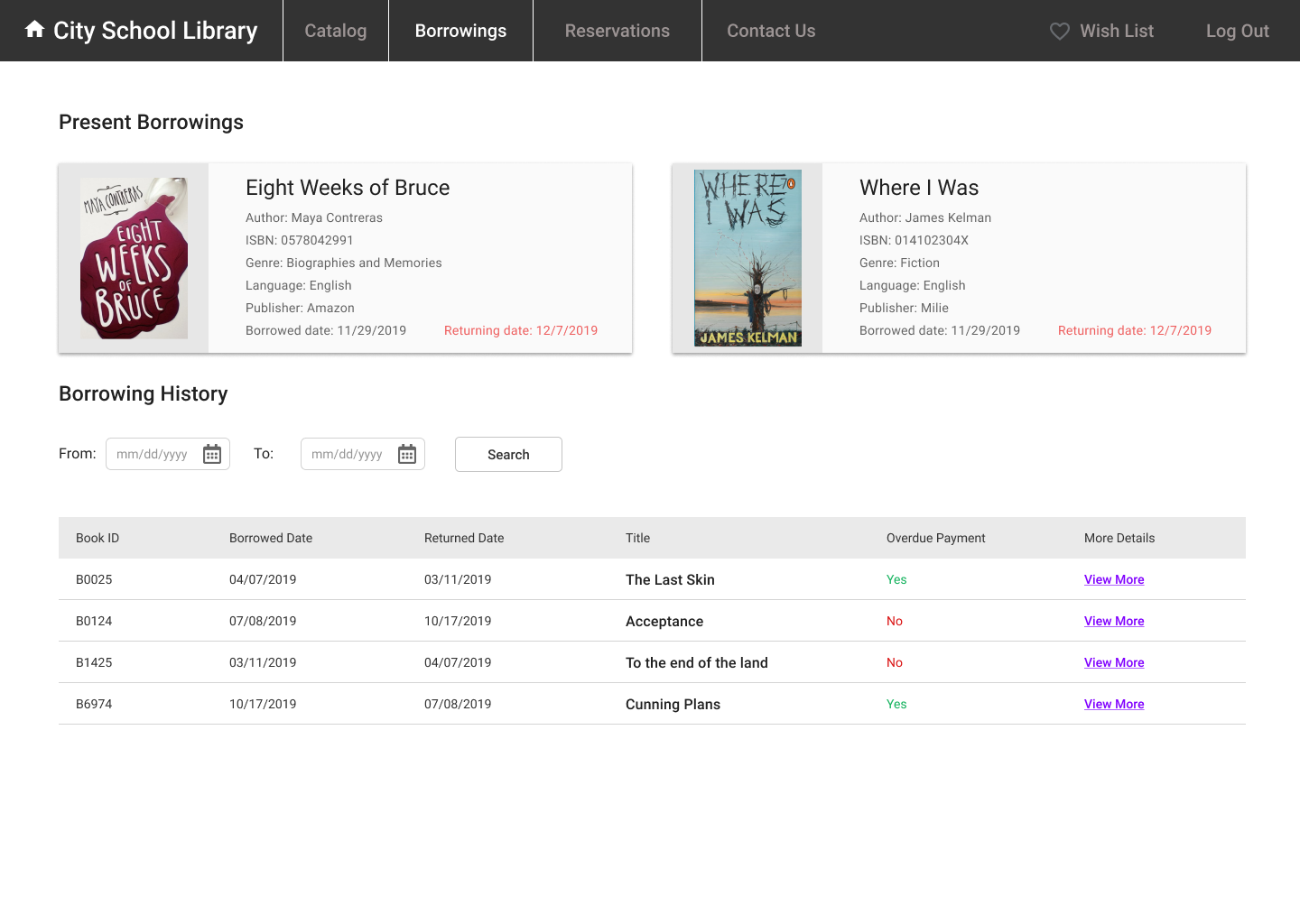


Figure 7: Borrowing page for see current borrowings and history

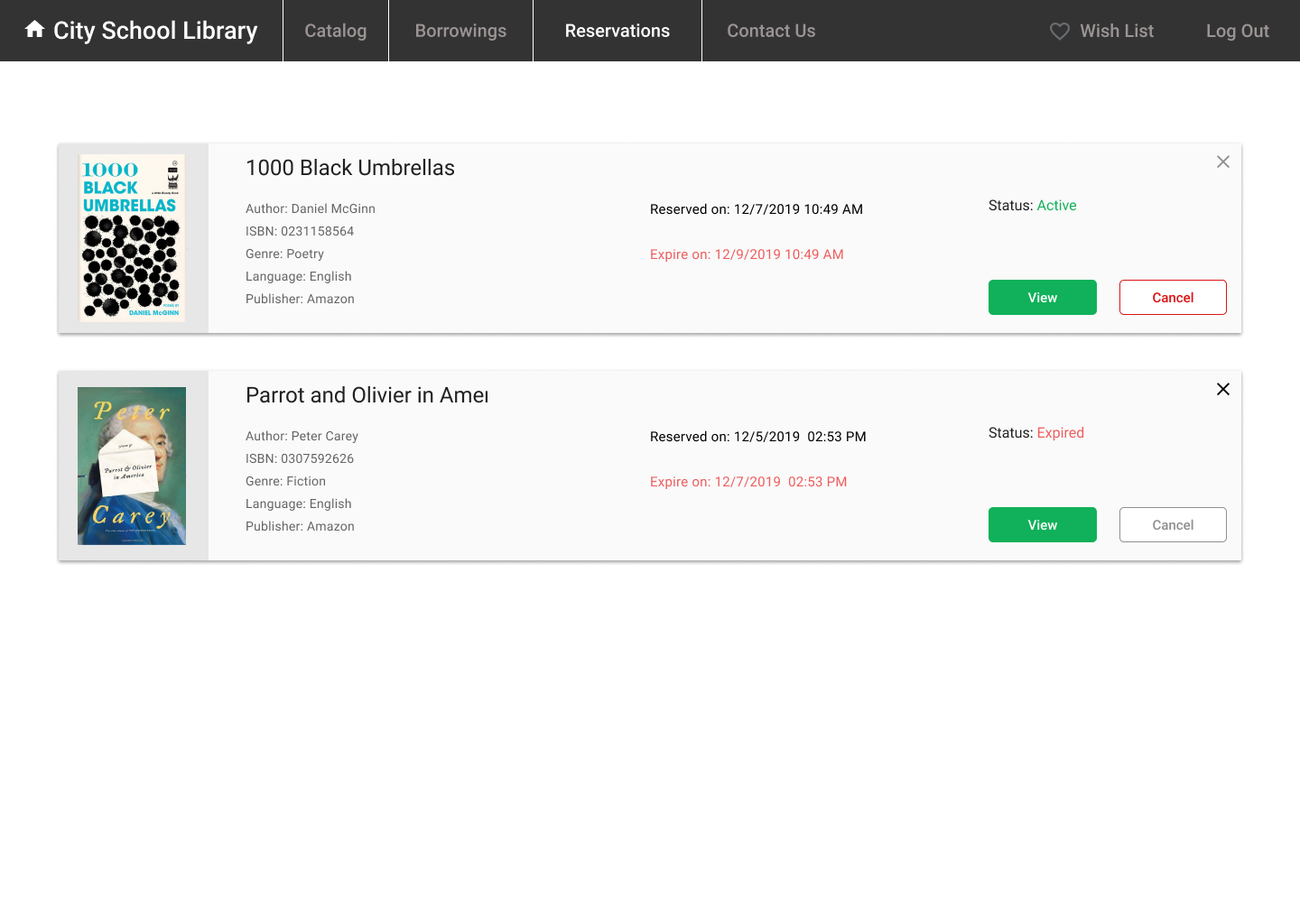


Figure 8: Student’s reservation page



Figure 9: Library contact details page

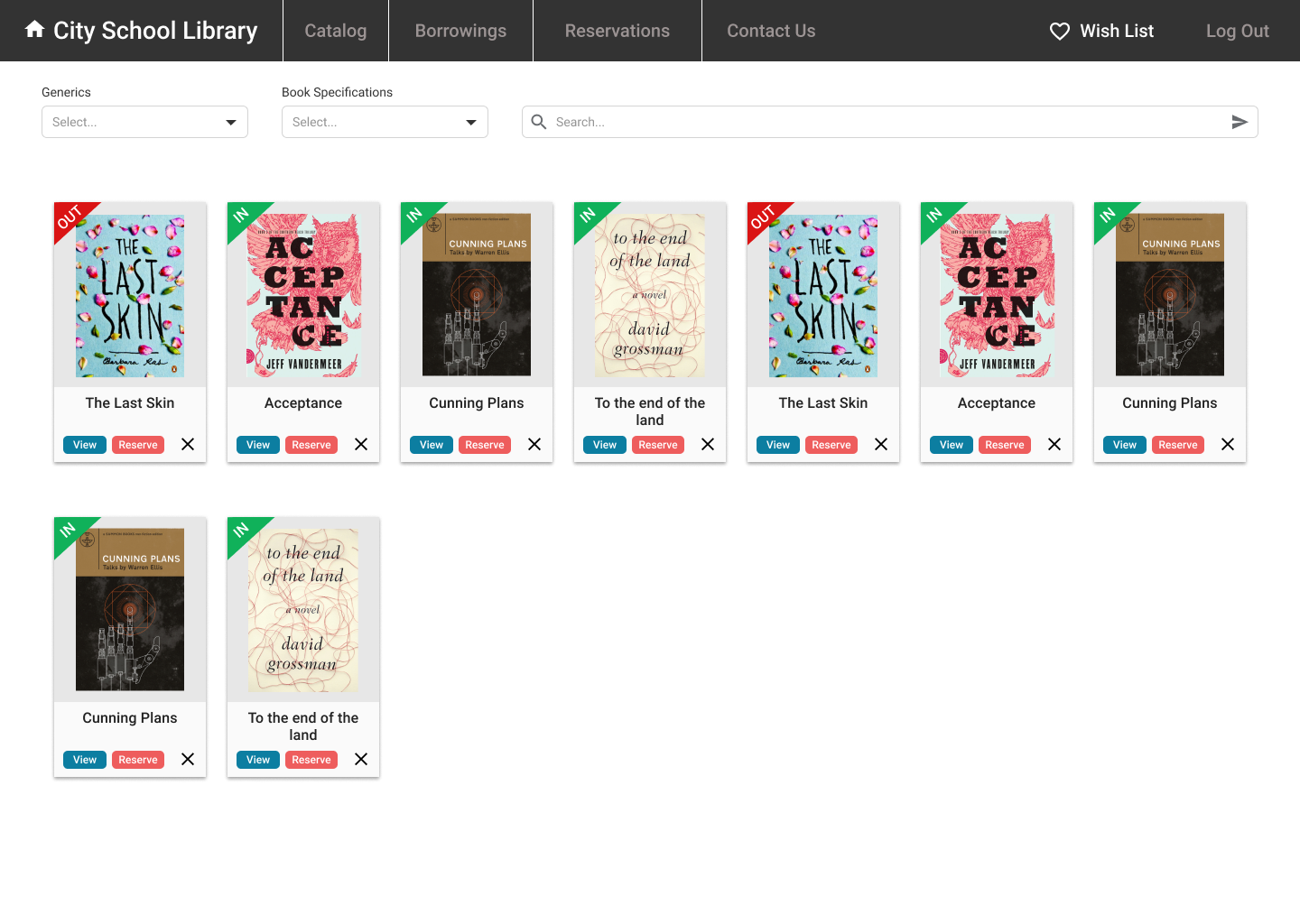


Figure 10: Student’s wish list page

**Librarian’s Sample UI**

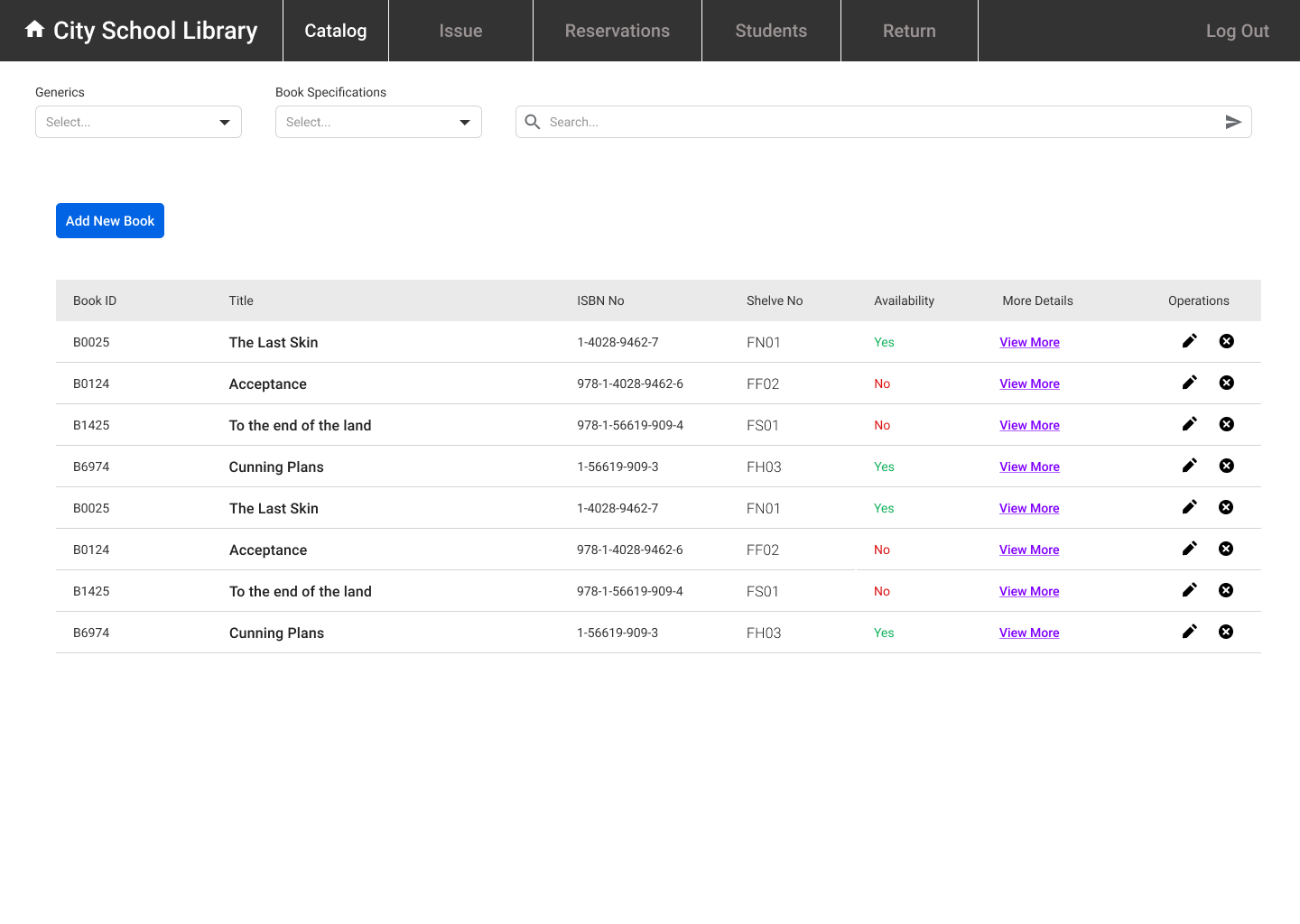


Figure 11: Librarian’s catalog page for manage books

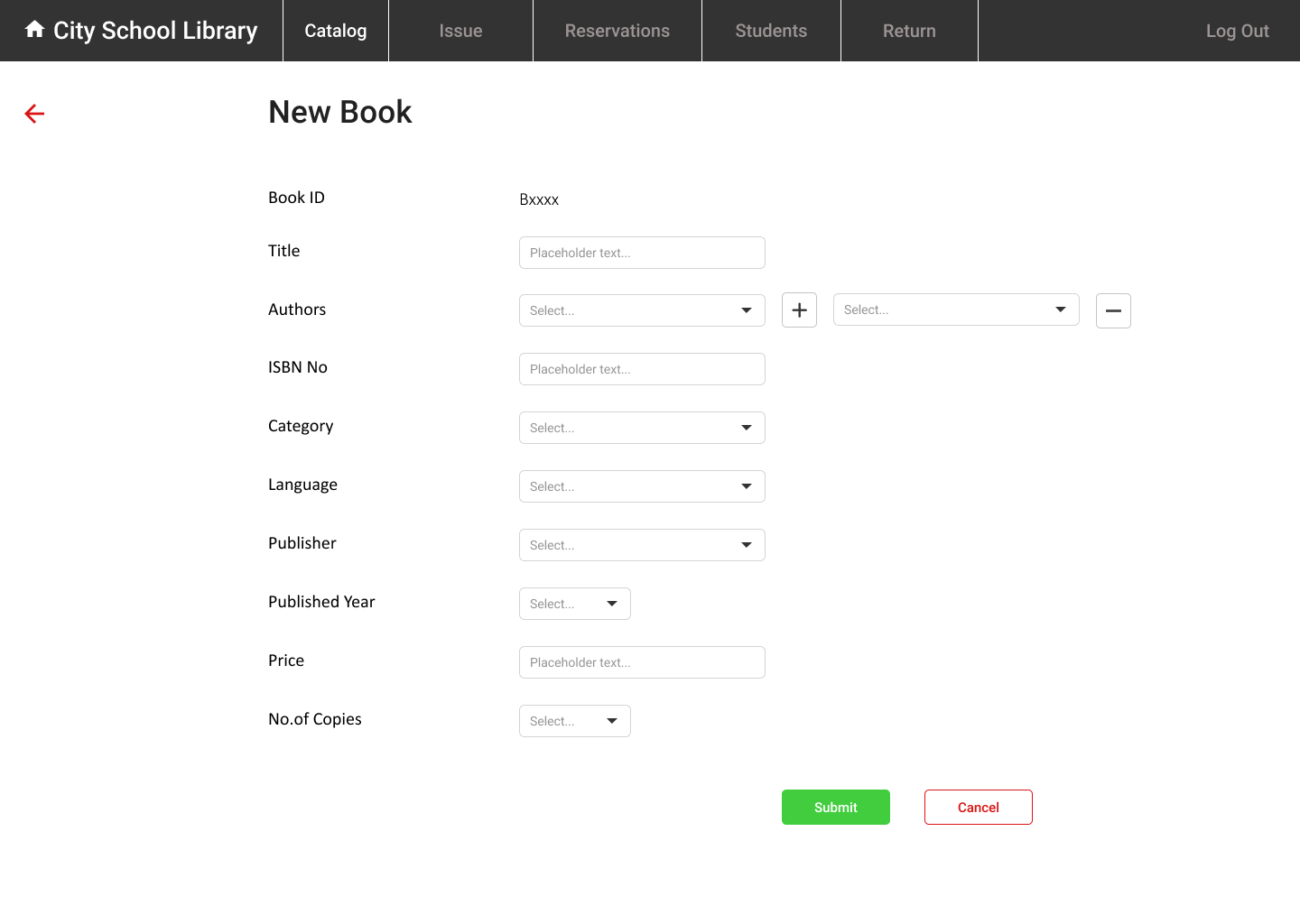


Figure 12: Insert a new book page

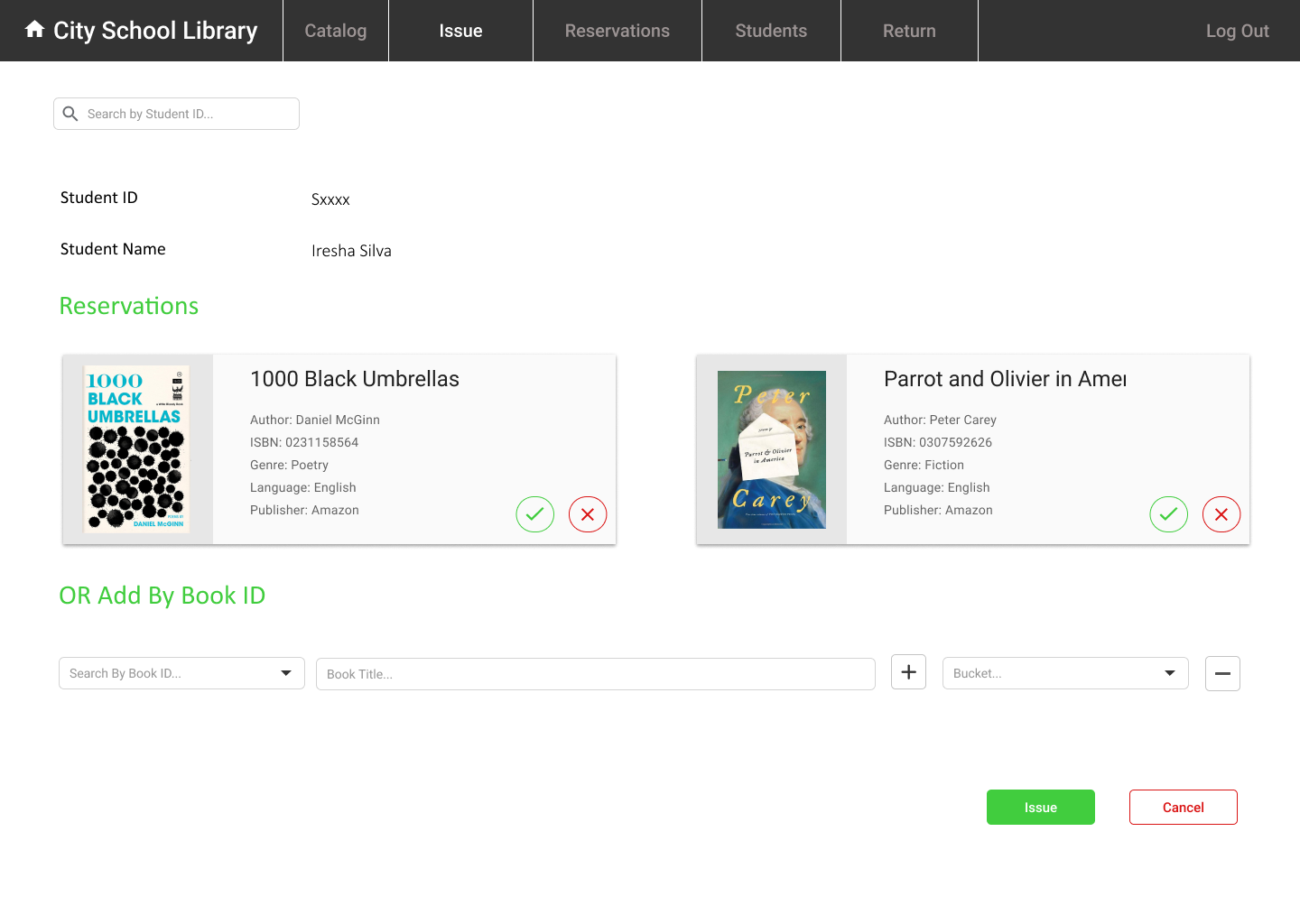


Figure 13: Librarian’s view when issuing books

Figure 14: Librarian’s view for all reservations of students

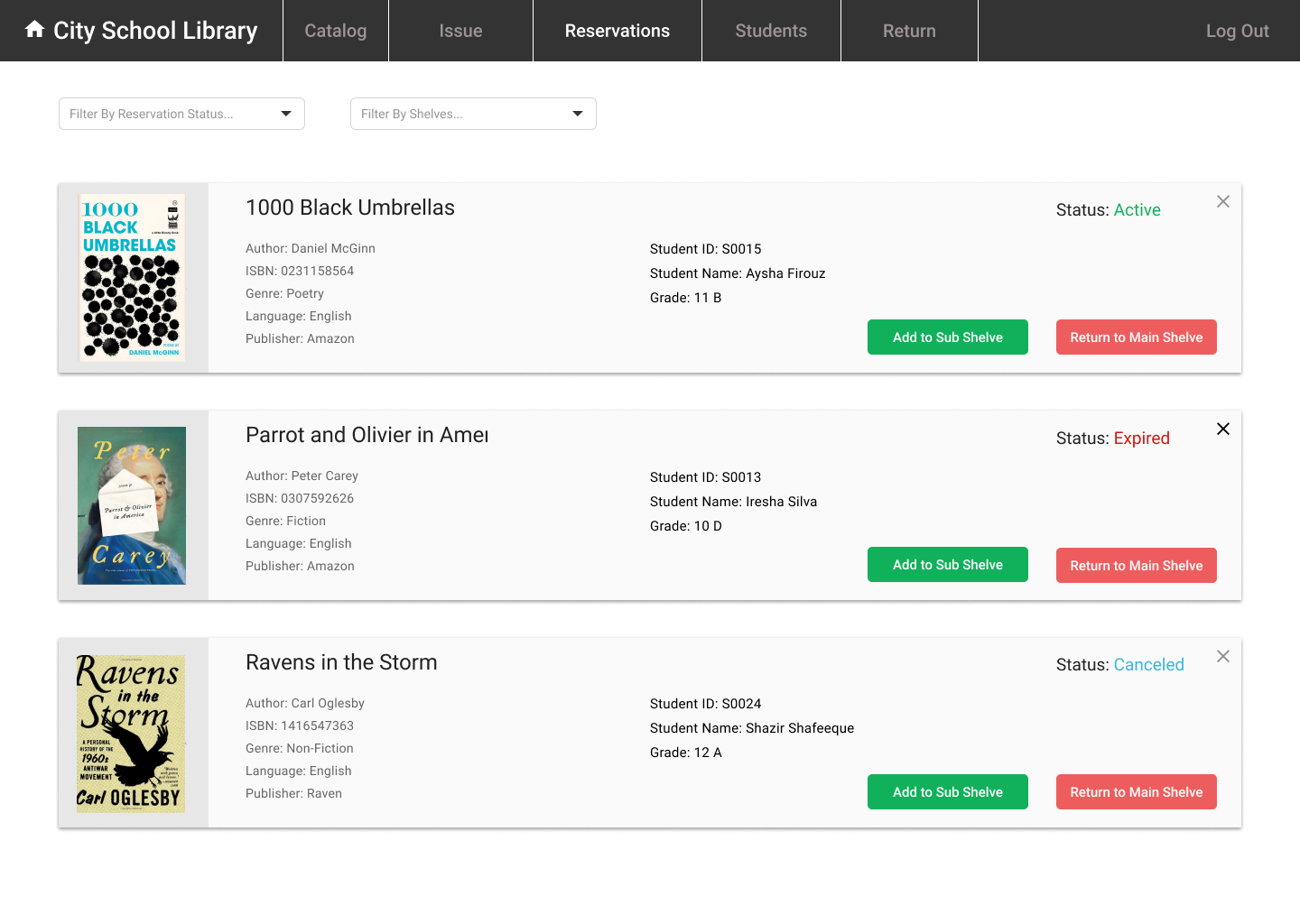




Figure 15: Librarian’s page for manage students

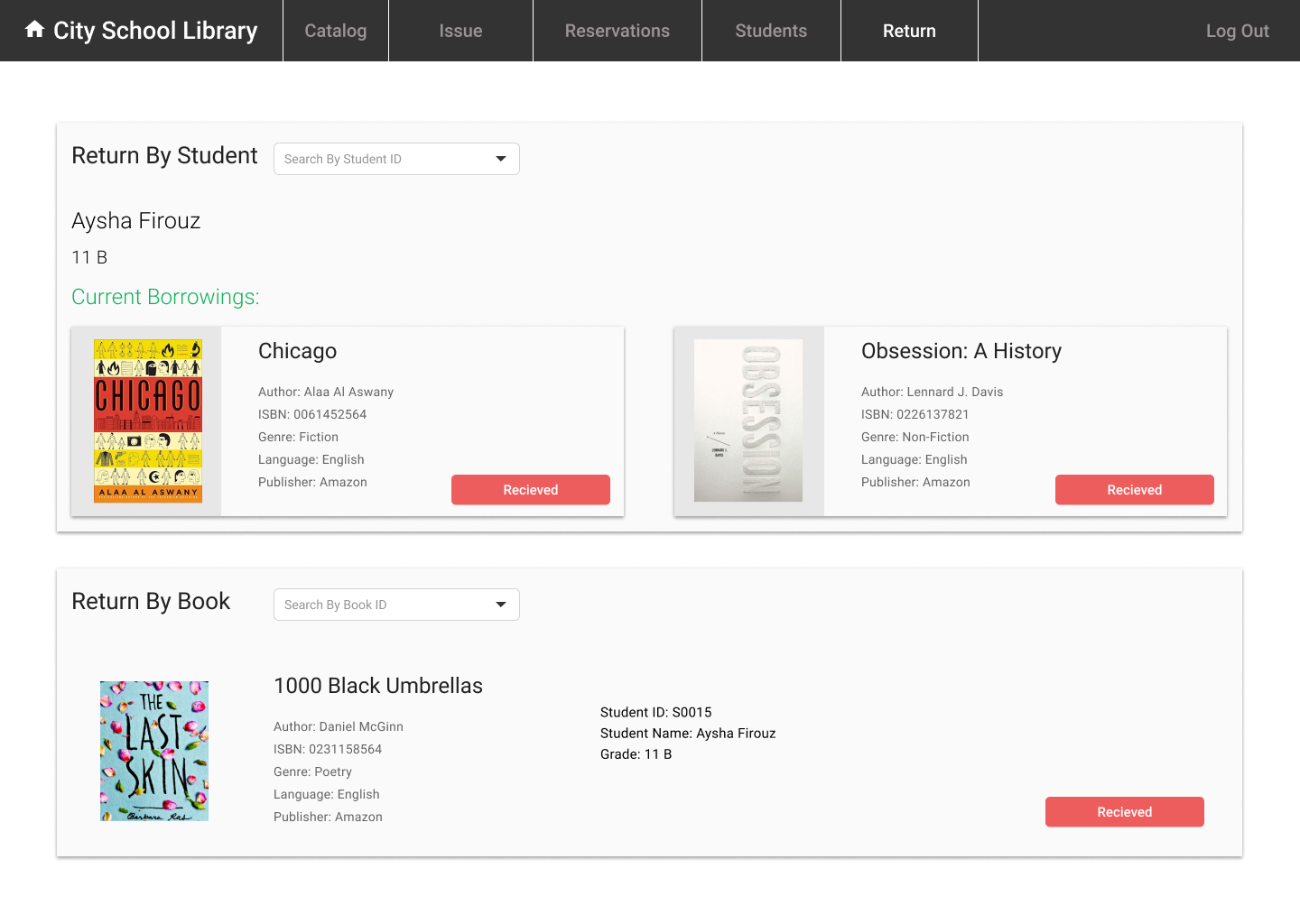


Figure 16: Librarian’s view when returning books

## Database relationship (ER) diagram

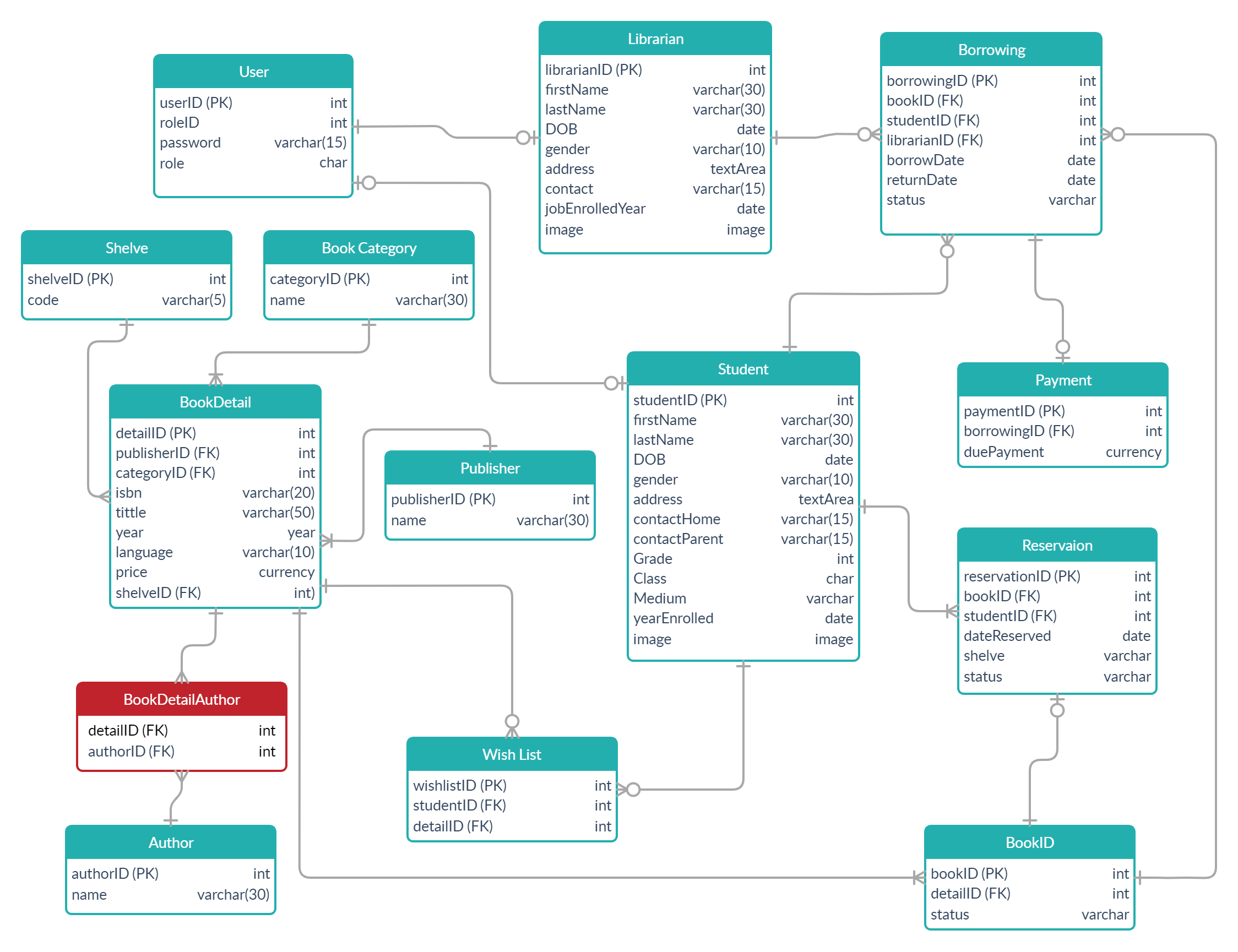


Figure 17: Entity-Relationship diagram of Library Management System

## Business logic

A student can reserve or borrow only two books at a time.

* Can’t reserve two books and borrow two other books.

A reservation is active only for 24 hours.

* Reservation automatically  expires after 24 hours.
* A student can cancel reservations before it expires.

48 hours of reservation history is shown.

* After 48 hours of a reservation, it automatically get removed from the database.

The student should return the book within 14 days.

* If he fails, a fine would be charged per day.

## Backend End Points

**Book Details**

**GET**

**/api/book**

Get all books

**GET**

**/api/book/{id}**

Get books by book ID

**GET**

**/api/borrowing/{book detail id}/availability**

Check book availability {return: IN or OUT}

**GET**

**/api/book/{Generics}/{Book Specification}**

Search book by {Generics} and {Book Specification}

**POST**

**/api/book**

Create a new book

**PUT**

**/api/book/{id}**

Update a book by book ID

**DELETE**

**/api/book/{id}**

Delete a book by book ID

**Students**

**GET**

**/api/student**

Get all students

**GET**

**/api/student/{id}**

Get student by ID

**GET**

**/api/student/{Attributes}**

Search student by {Attributes}

**PUT**

**/api/student/{id}/block**

Block a student by student ID

**PUT**

**/api/student/{id}/reset**

Reset password of a student by student ID

**Reservation**

**GET**

**/api/reservation**

Get all reservations

**GET**

**/api/reservation/{id}**

Get reservation by student ID with Book Details

**GET**

**/api/reservation/{status}**

Get reservation by status

**GET**

**/api/reservation/{shelve}/shelve**

Get reservations by shelve

**PUT**

**/api/reservation/cancel**

Cancel the reservation record

**PUT**

**/api/reservation/{id}/add**

Update a reservation shelve as added to sub shelve

**PUT**

**/api/reservation/{id}/return**

Update a reservation shelve as returned to main shelve

**POST**

**/api/reservation**

Create reservations for students if possible(maximum reservations per student=2)

**Borrowing**

**GET**

**/api/borrowing/{id}**

Get current borrowings of a student by student ID,Check if there are any overdue payments

**GET**

**/api/borrowing/{from date}/{to date}**

Get borrowings between dates

**GET**

**/api/borrowing/{book id}/book**

Get borrowings record by book id

**PUT**

**/api/borrowing/{borrowing id}**

Update the status of borrowing as “returned ” borrowing ID,Update the return date

**POST**

**/api/borrowing/{id}**

Create a new borrowing with(status:borrowing){id specifies the number of books issued at a time ex: min=1 or max=2 }

**Payment**

**GET**

**/api/payments/{id}**

Get all payments of a student by student ID

**POST**

**/api/payment**

Create a new payment by borrowing ID

**Wish list**

**POST**

**/api/wishlist**

Create a new wishlist for a student with book detail ID

**DELETE**

**/api/wishlist/{id}**

Delete a wishlist by wishlist ID

**Publisher**

**GET**

**/api/publisher**

Get all publishers

**GET**

**/api/publisher/{id}**

Get all publisher by ID

**POST**

**/api/publisher**

Create a new publisher

**PUT**

**/api/publisher/{update}/{id}**

Update a publisher by publisher ID

**DELETE**

**/api/publisher/delete/{id}**

Delete a publisher by ID

**Author**

**GET**

**/api/author**

Get all authors

**GET**

**/api/author/{id}**

Get all author by ID

**PUT**

**/api/author/{id}**

Update a author by author ID

**POST**

**/api/author/{id}**

Create a new author

**DELETE**

**/api/author/{id}**

Delete a author by author ID

**Shelve**

**GET**

**/api/shelve**

Get all shelve

**GET**

**/api/shelve/{id}**

Get shelve by ID

**DELETE**

**/api/shelve/{id}**

Delete a shelve by shelve ID

**PUT**

**/api/shelve/{id}**

Update a shelve by shelve ID

**POST**

**/api/shelve**

Create a new shelve

**Librarian**

**GET**

**/api/librarian**

Get all librarian

**GET**

**/api/librarian/{id}**

Get librarian by ID

**POST**

**/api/librarian/new**

Create a new librarian

**PUT**

**/api/librarian/update/{id}**

Update a publisher by librarian ID

**PUT**

**/api/librarian/reset/{id}**

Reset password of a librarian by librarian ID

**DELETE**

**/api/librarian/delete/{id}**

Delete librarian by librarian ID

**User**

**GET**

**/api/user**

Get all users

**GET**

**/api/user/{id}**

Get user by ID

**POST**

**/api/user/new**

Create a new user

**PUT**

**/api/user/update/{id}**

Update user by user ID

**PUT**

**/api/user/reset/{id}**

Reset password of a user by user ID

**DELETE**

**/api/user/delete/{id}**

Delete user by user ID

**Genre**

**GET**

**/api/genre**

Get all genre

**GET**

**/api/genre/{id}**

Get genre by ID

**POST**

**/api/genre/new**

Create a new genre

**PUT**

**/api/genre/update/{id}**

Update genre by genre ID

**DELETE**

**/api/genre/delete/{id}**

Delete genre by genre ID

## Final screenshots