Application Requirements Specification

For

<Library Management System>

Prepared By:

<Haoyu Hu, Li Fu, Taixu Jin>

**Purpose and Scope Statement**

The purpose of this project is to provide a centralized platform for library administrators to manage books and students to use library services such as search, borrow and return basic services. The system will allow administrators to add, delete, and update book records, and the system will also allow students use basic library services such as search, borrow and return.

The scope of project is mainly three parts, one is overall UI interactive interface, including register, login and the other subfunction user interface, the other is overall interaction logic implementation of the entire back-end program, the last is database for implementing search, add, delete and update functions.

**Requirements Narrative**

In this project, this application has two user roles: librarian and student.

When a librarian account is created, they can log in through the librarian user interface. The librarian user interface includes the following functionalities:

1. Add Book Information: Librarians can add and book information, including book number, location, and borrowing information.
2. Search Book Information: Librarians can search for book information using various criteria, such as book title, author, or book number.
3. Delete book information: Librarians have the capability to delete book information by specifying the book number.
4. Update Book Information: Librarians can update book information by specifying the book number and updating relevant fields as listed in the table as item 1.

When a student user account is created, they can log in through the user interface designed for students. The student user interface includes the following functionalities:

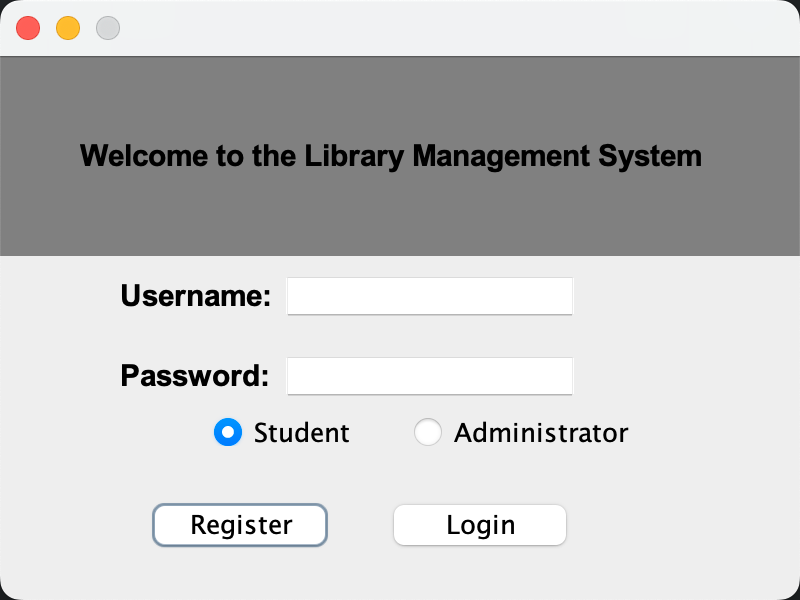
1. Overview of Book Information: Students can view a summary of the available book information in the library.
2. Book Retrieval Function: Students can search for books using criteria such as title, author, or book number.
3. Borrow book: Students can borrow books from library if books available.
4. Return book: Students can return books and then update the book borrow status.

Additionally, the application provides user registration and login interfaces:

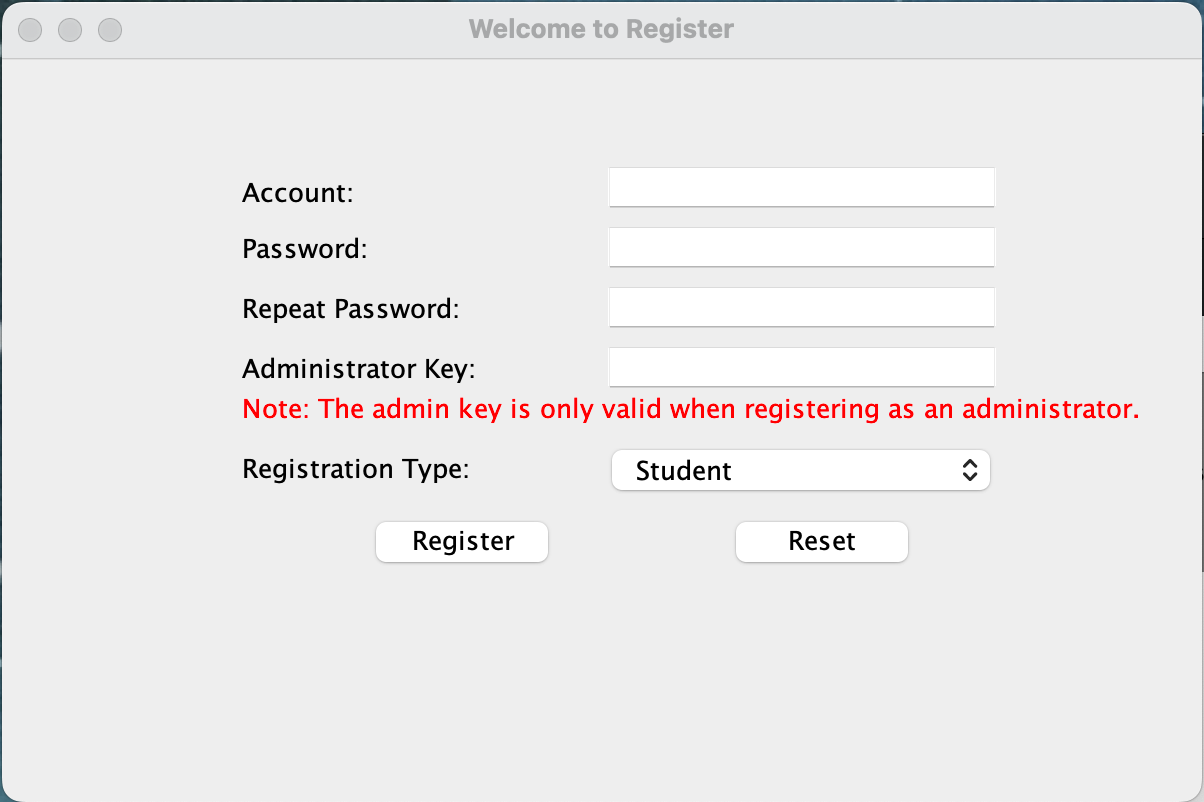
1. Registration Page: Both librarians and students can register by selecting their user category (librarian or students) and providing information such as name, username, password, etc.
2. Login Page: Users can log in by selecting their role and entering their username and password.

**Objectives**

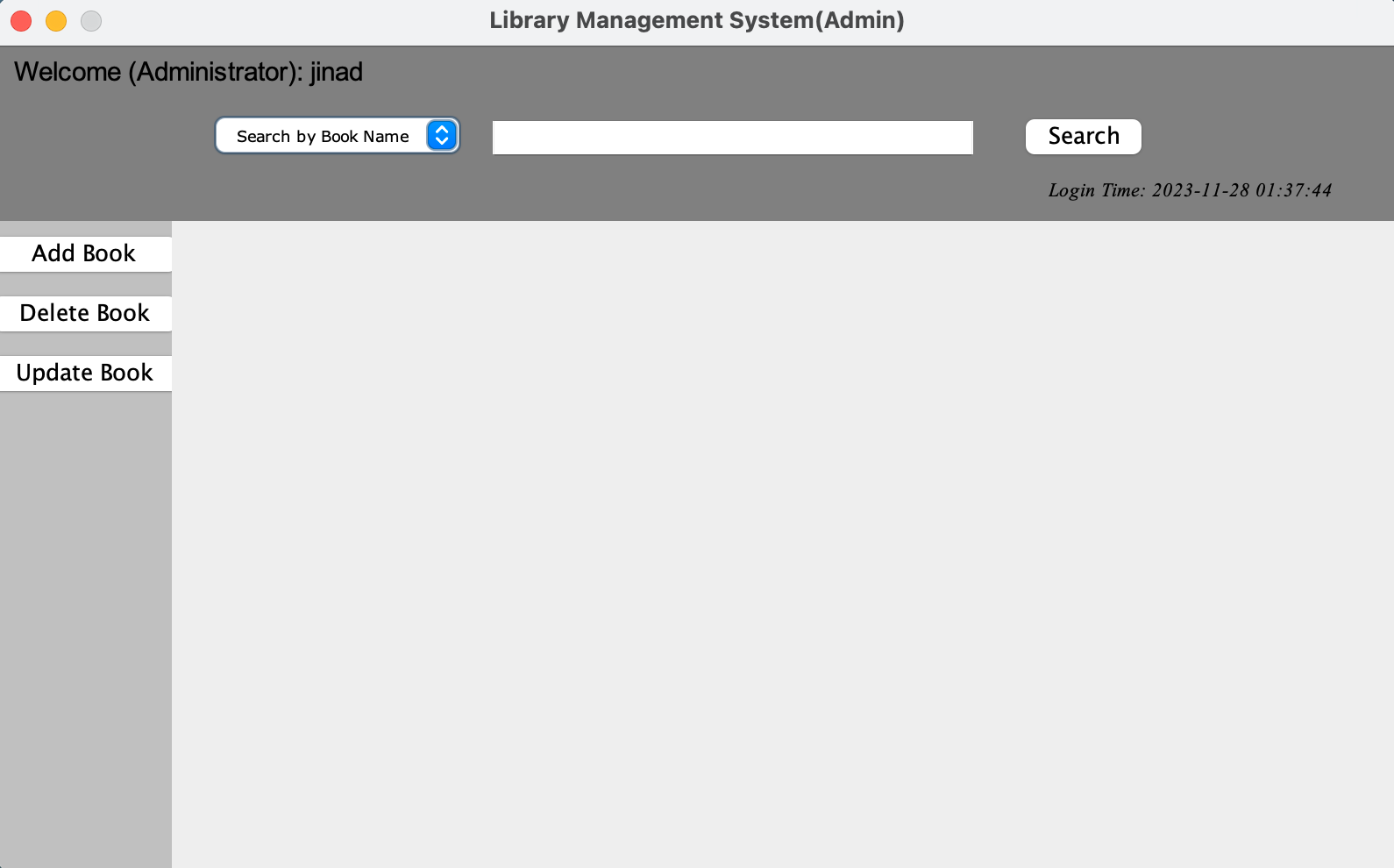
The Login screen



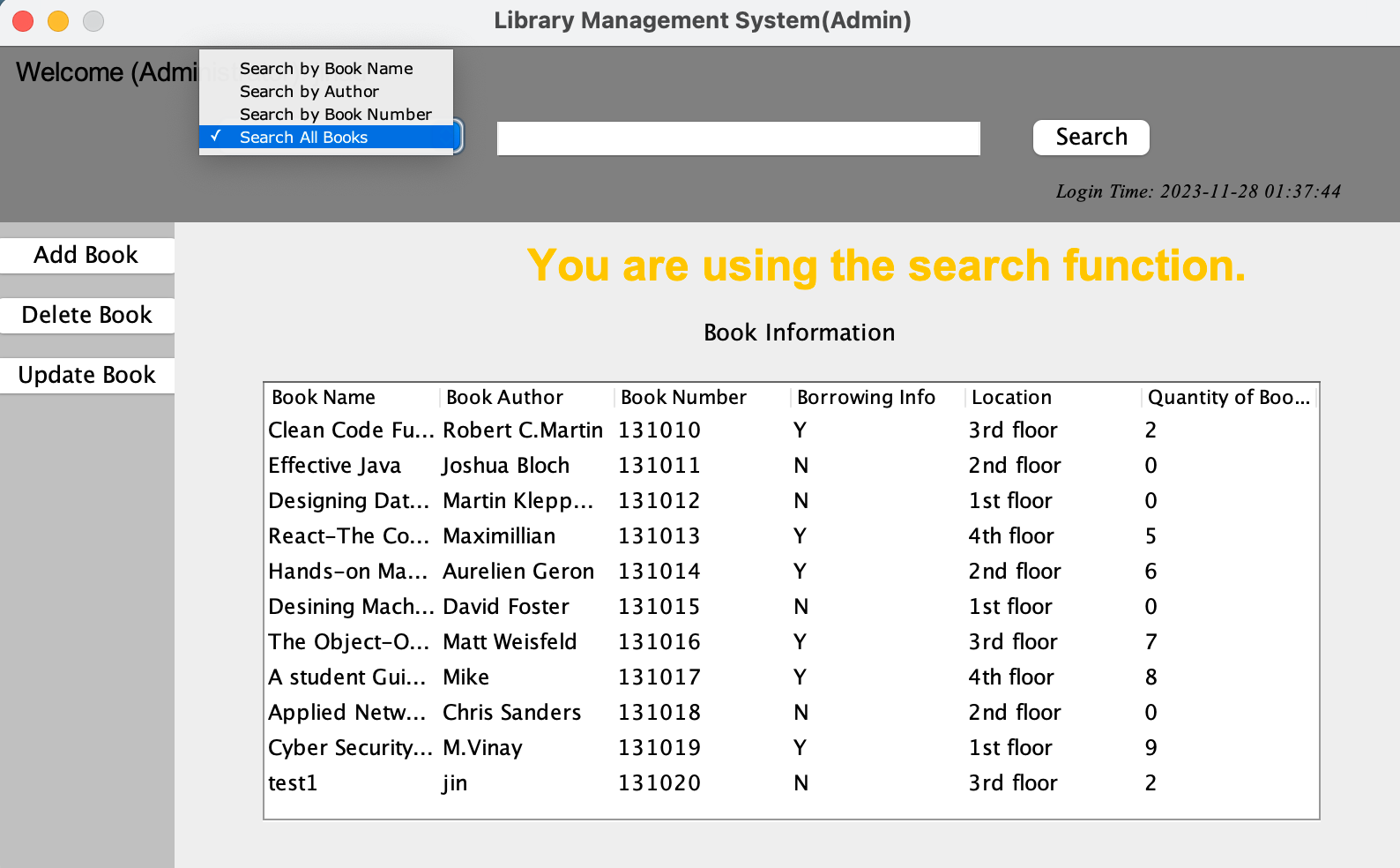
The registration screen



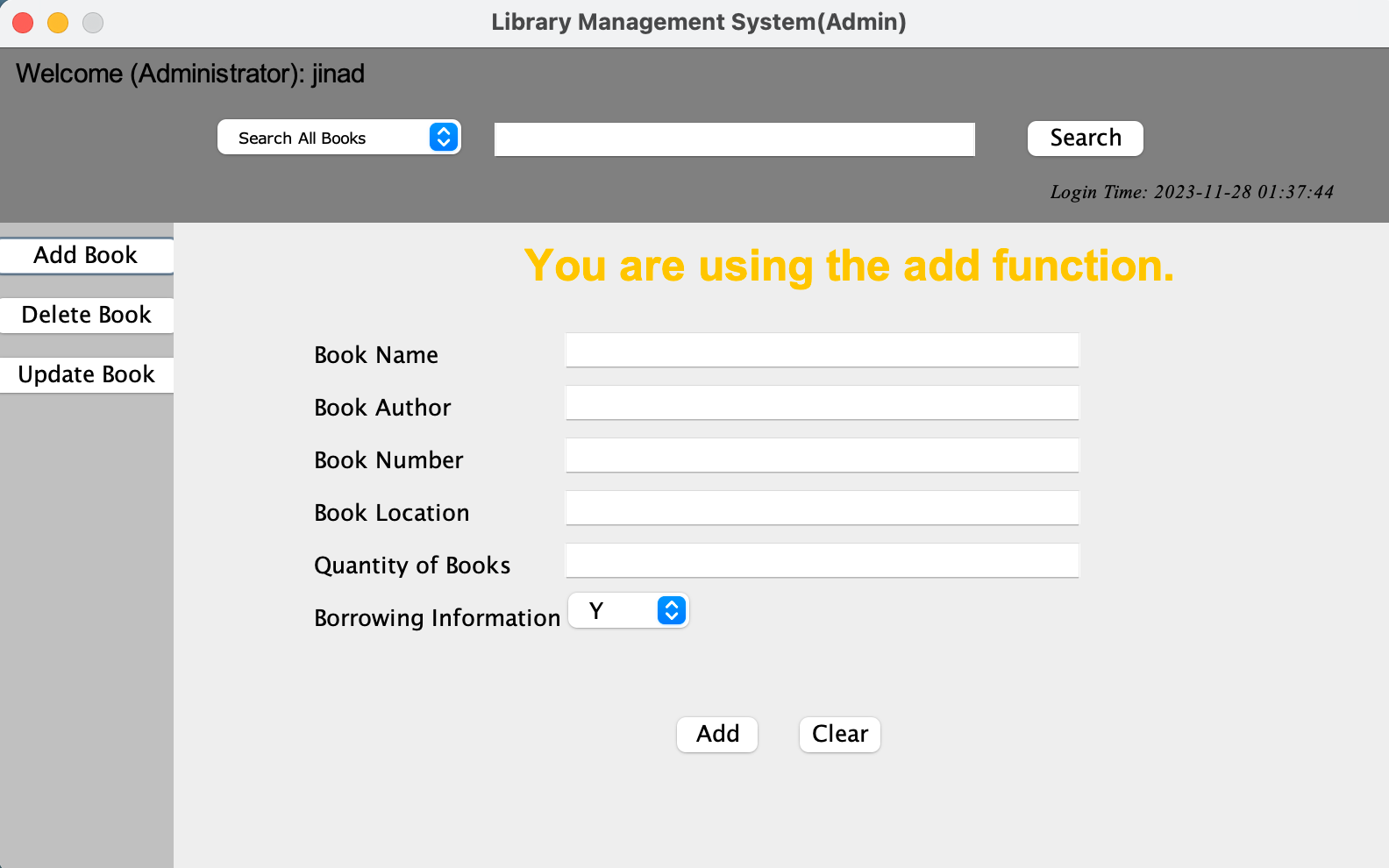
3. The screen of administration user



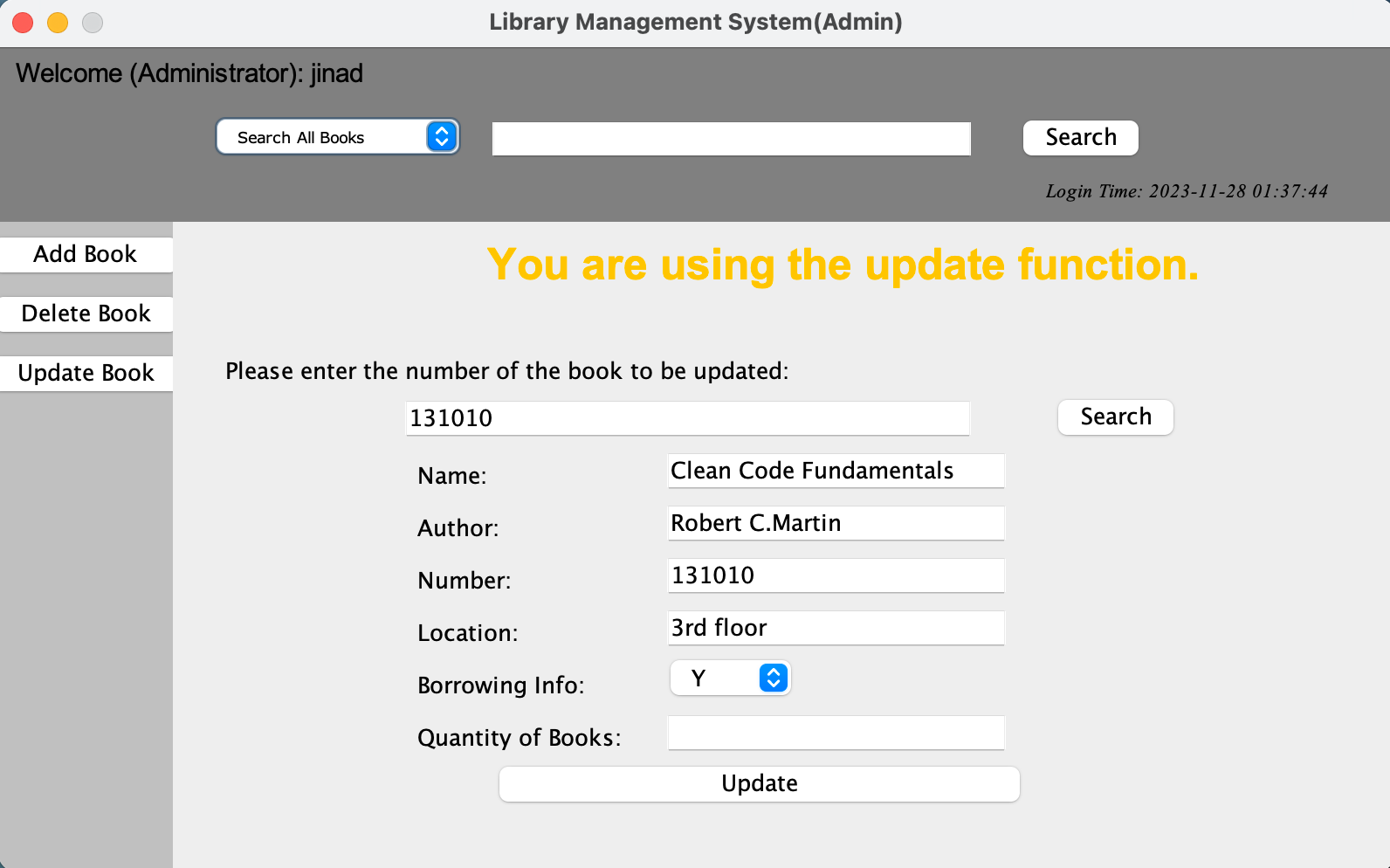
1. The screen of administration search function



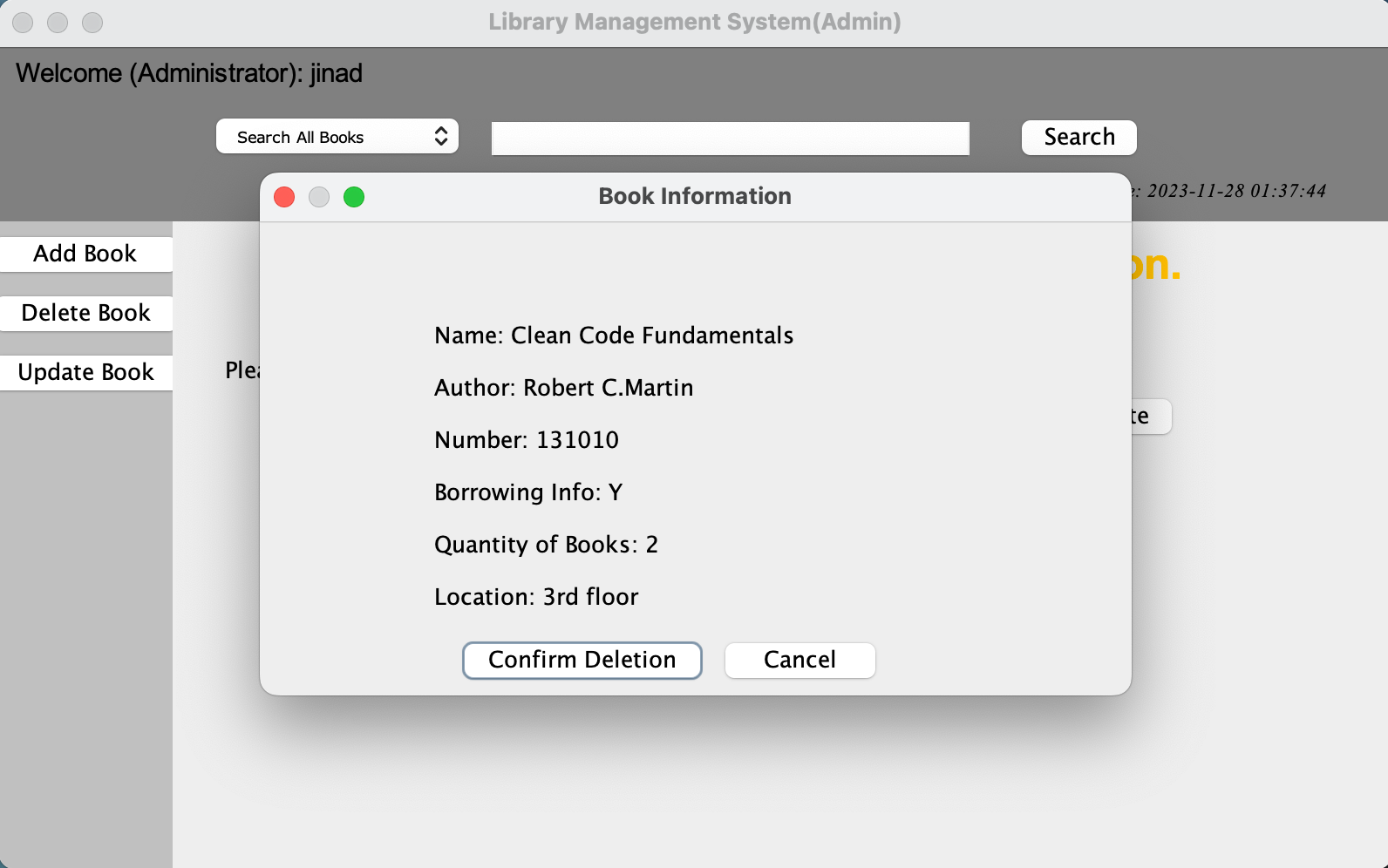
1. The screen of administration add function



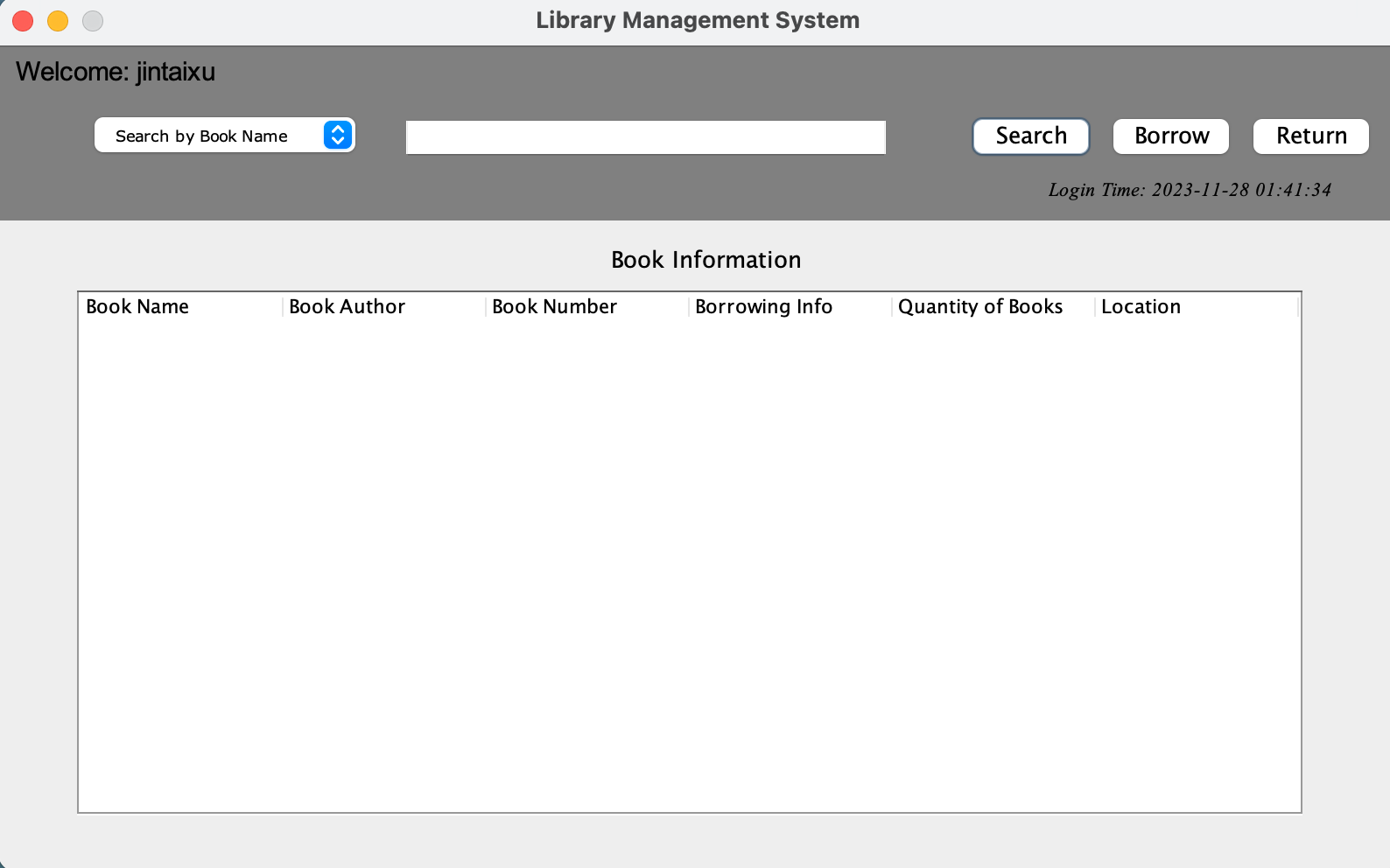
1. The screen of administration update function



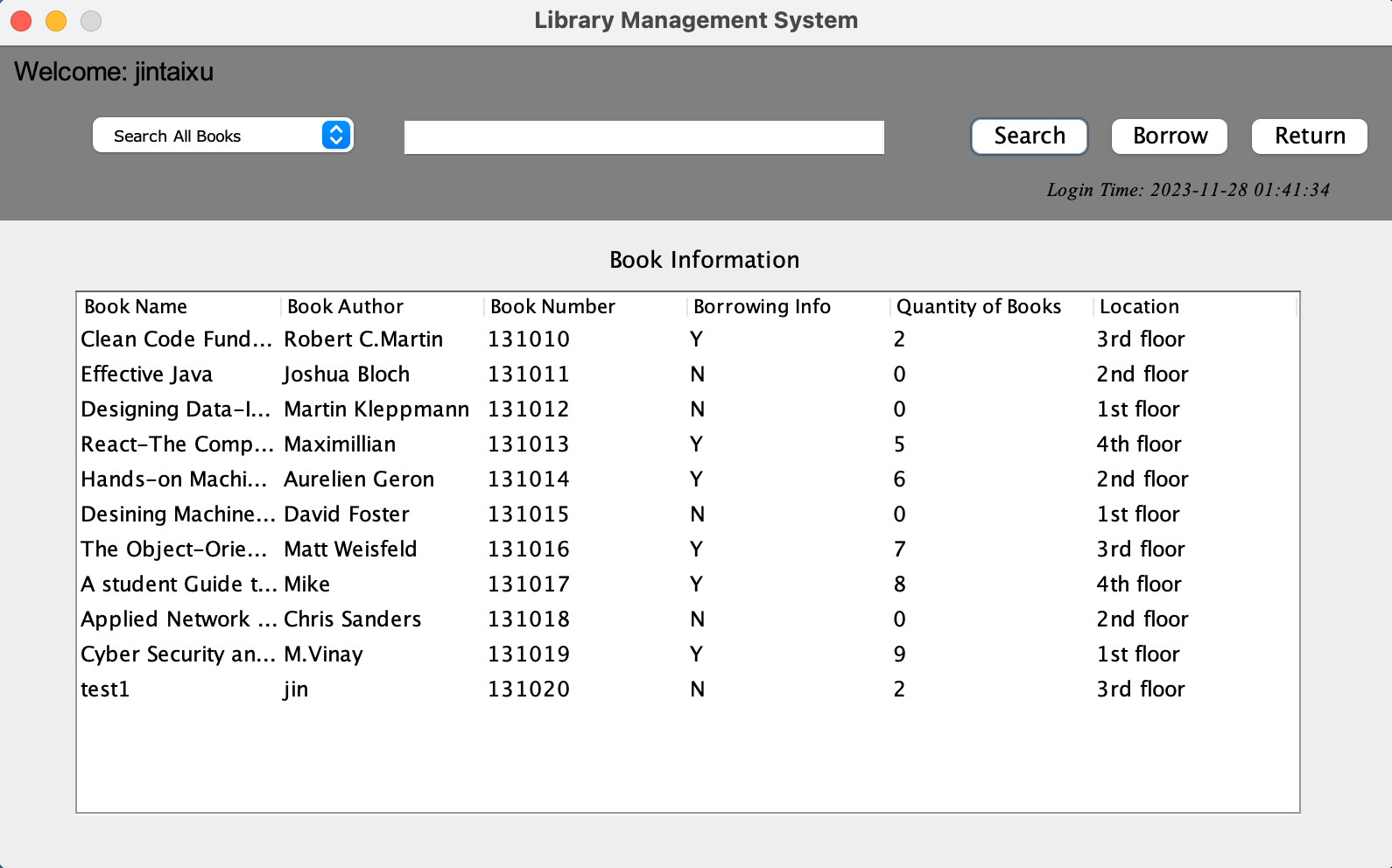
1. The screen of administration delete function



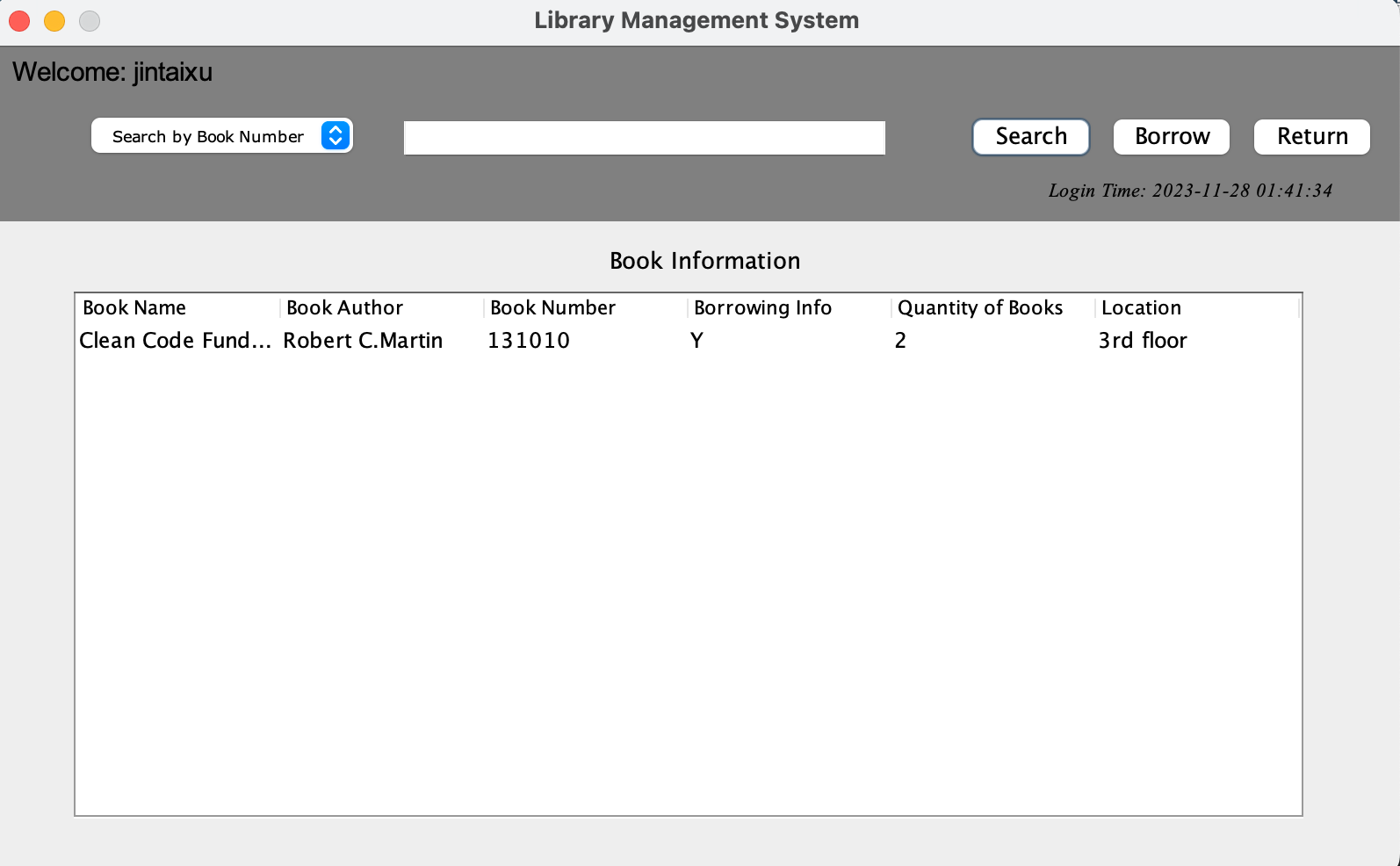
1. The screen of student user



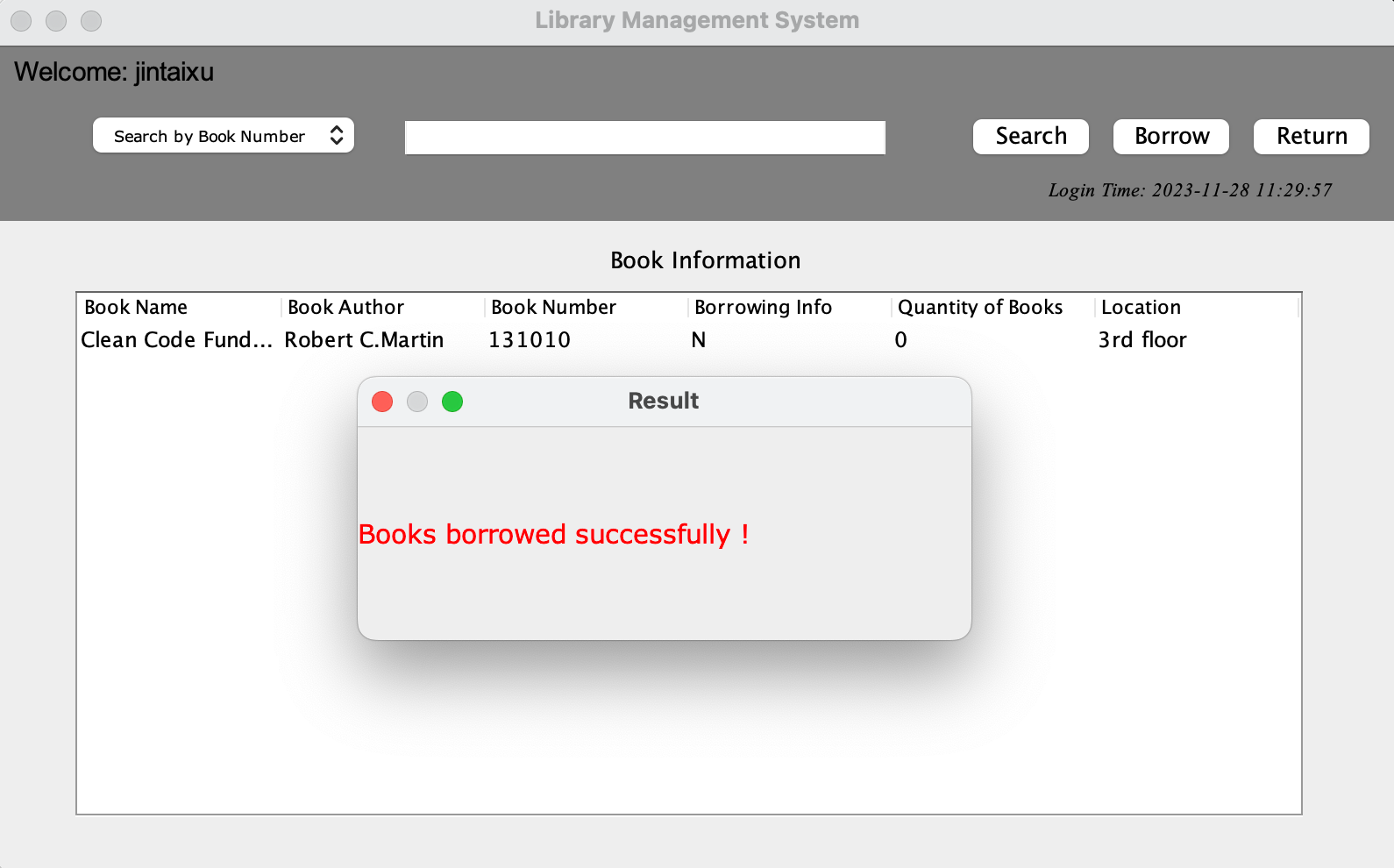
1. The overview function screen of student user



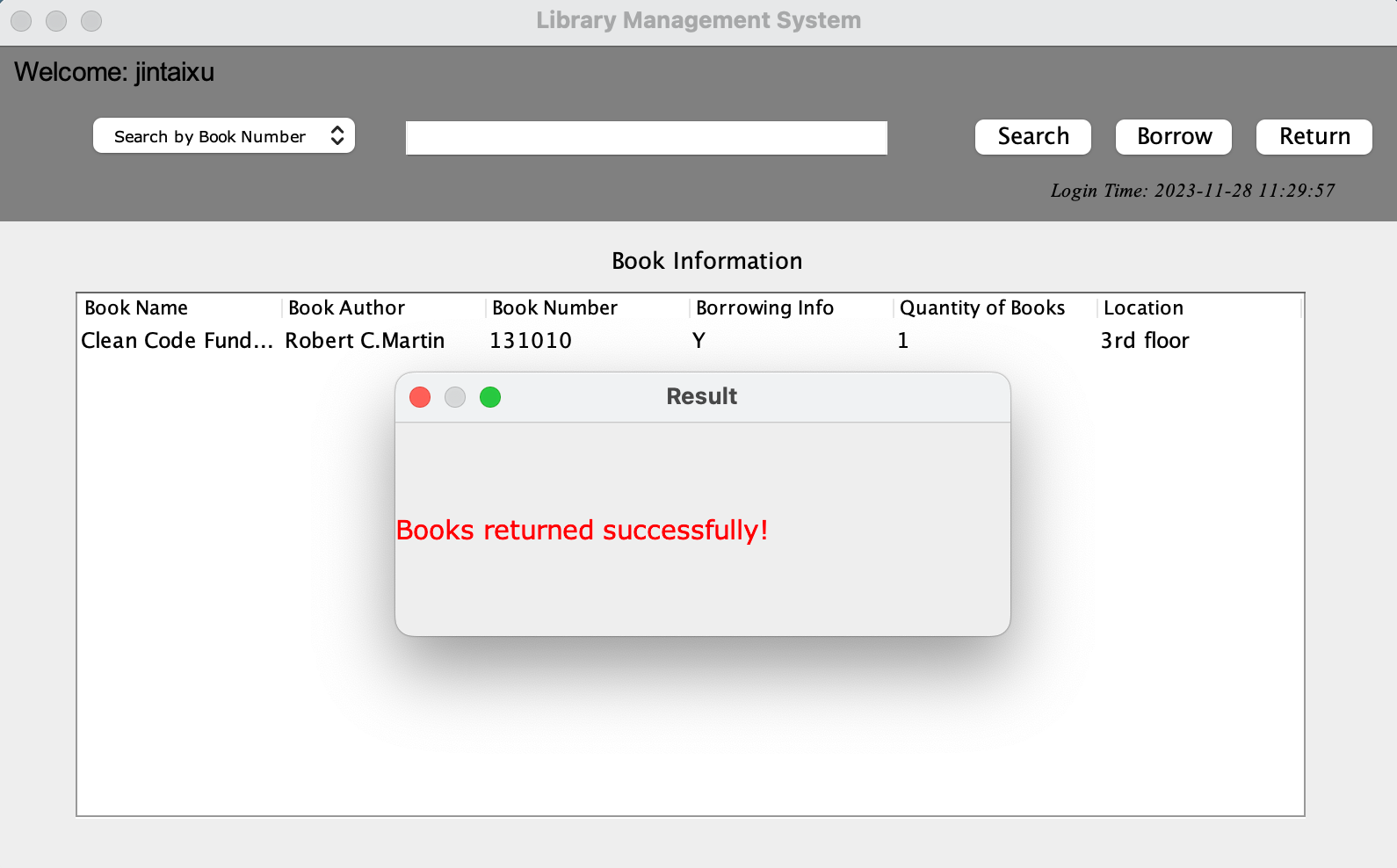
1. The book retrieval function of student user



1. The book borrow function of student user



1. The book return function of student user



**Functional Specification**

The overall programming follows MVC design pattern, so in this project we divide into five parts, and we will descript detail as below.

For data access object file

This file represents the data and the business logic of the application. It is responsible for accessing the data layer, retrieving data, processing it, and sending it back to the view or updating the database as needed.

AdminDao.java

RegistDao.java

UserDao.java

For model file

This file creates database attributes objects for easily calling the functions.

Admin.java

AdminCode.java

Book.java

Register.java

User.java

For service file

This file acts as an intermediary between the model and the UI. It listens to the input from the view, processes the user's data (with the help of the model), and returns the output display to the view.

AdminService.java

RegistService.java

UserService.java

For UI file

This file is the user interface of the application. It displays the data (from the model) to the user and sends user commands (like button clicks, form submissions) to the controller.

AdminView.java

Index.java

Login.java

RegisterDialog.java

UserView.java

For utils

This file is to complete the database connection and some custom classes.

JDBCUtil.java

Tools.java

Technologies Needed

Java

Java Swing gui for UI

Database – MySQL database

**Logic Specification**

A diagram of a student

Description automatically generated

The project combines Java Swing with MVC which likely provides a robust framework for creating a user-friendly interface while maintaining a clean separation of the application's data processing and user interaction logic. This approach is particularly beneficial for complex applications requiring a high degree of organization, maintainability, and scalability.

**Java Swing:**

1. Rich User Interface: Swing provides a comprehensive set of GUI components and allows for the creation of a more sophisticated and visually appealing user interface.
2. Platform Independence: Being part of Java, Swing applications are platform-independent, which means they can run on any operating system that supports Java.
3. Customizability: Swing components are highly customizable, allowing developers to create a unique look and feel for their applications.
4. Integration with MVC: Swing naturally supports the MVC pattern, making it easier to implement a clean separation of concerns, which is beneficial for the maintainability and scalability of the application.

**MVC Pattern:**

1. Organized Code Structure: MVC organizes the code in a way that separates the application's logic, UI, and data management, which enhances code readability and maintainability.
2. Ease of Modification: Because the presentation and business logic are separated, changes in the user interface do not affect the data handling, and vice versa. This makes the application more adaptable and easier to update or modify.
3. Facilitates Team Development: Different teams can work on the Model, View, and Controller independently, improving development efficiency.
4. Reusability and Scalability: Models, being separated from Views and Controllers, can be reused across different parts of the application. Also, MVC supports scalability as the application grows in complexity.