INTERNATIONAL UNIVERSITY

VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY

School of Computer Science and Engineering

PROJECT REPORT

Library Management System

For Web Application Development COURSE (IT093IU)

Course by
Assoc. Prof. Nguyen Van Sinh and MSc. Nguyen Trung Nghia

By
Trần Thiên Phú - ITITIU22124

Github: Link

TABLE OF CONTENTS

Contents

TABLE OF FIGURES

I.INTRODUCTION

I.Abstract

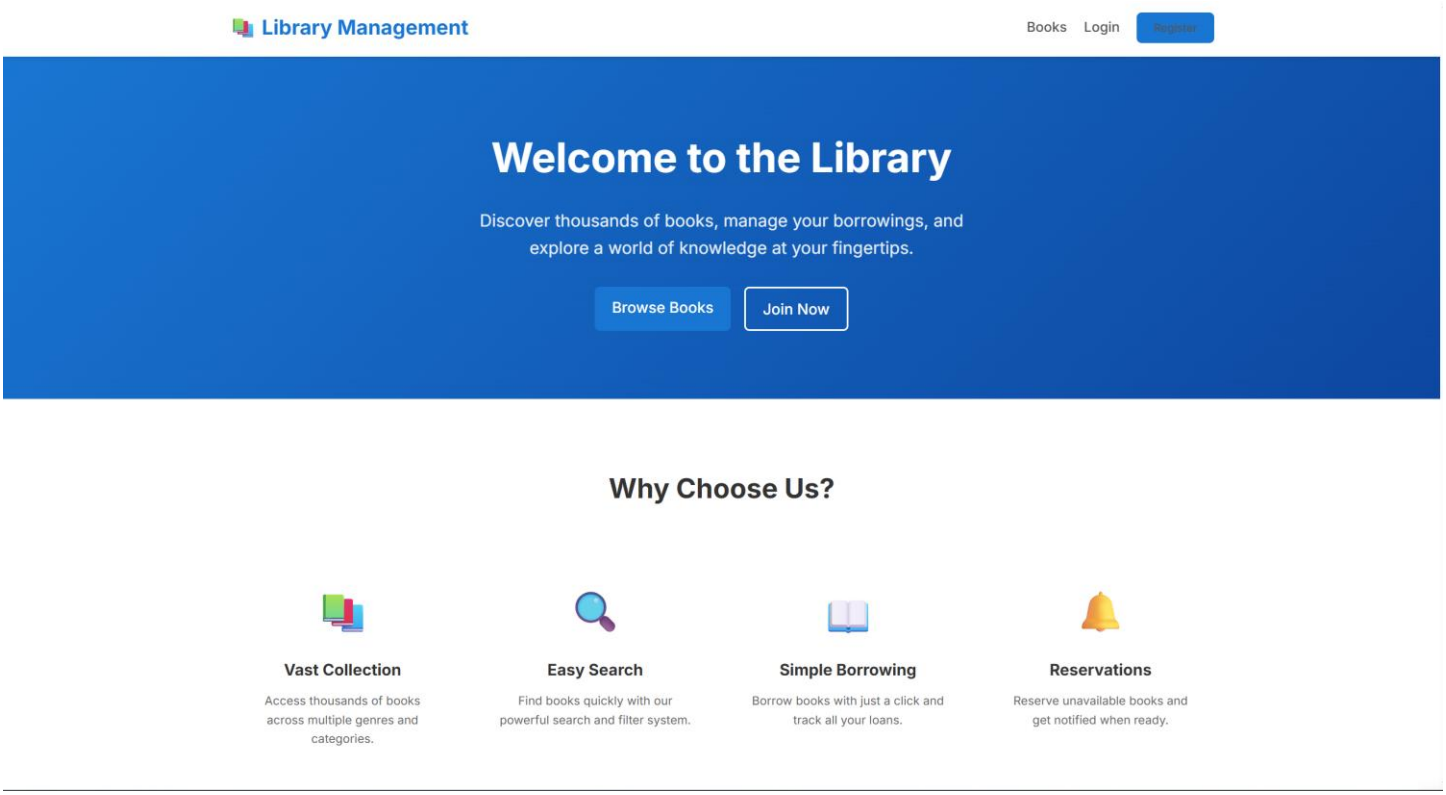

Figure 1 Abstraction

This system streamlines essential library functions including book cataloging, user registration, borrowing/returning processes, and inventory tracking to enhance efficiency over manual methods. Designed with object-oriented principles, it supports multiple user roles such as librarians and patrons, utilizing file-based or simple database persistence for data management. The public GitHub repository enables collaboration and version control, though it currently lacks published releases or packages.

Key Features

- User and Admin authentication/authorization
- Book management (CRUD)

- Borrowing and returning books
- Fine calculation and management
- Book reservations
- Email notifications for borrow, due, overdue, reservation, and return events
- Modern UI for users and admin

2. Overview

The development of the Library Management System web application exemplifies the transformative impact of modern web technologies on information management in educational and institutional settings. Web applications have become essential tools for libraries, enabling seamless access to resources, efficient management of collections, and enhanced user engagement through intuitive digital interfaces. By leveraging the ubiquity of web browsers, this system ensures that students, staff, and administrators can interact with library services from any device, at any time, without the need for specialized software.

This project harnesses the strengths of a full-stack architecture, combining a React.js frontend for dynamic, responsive user experiences with a robust Java Spring Boot backend that manages business logic, data persistence, and security. The system supports a wide range of library operations, including user registration, book catalog management, borrowing and returning processes, reservations, and fine management. Real-time notifications and role-based access control further enhance the system's functionality and security.

Throughout the development process, the team addressed key challenges inherent to web application design, such as ensuring data consistency, implementing secure authentication and authorization, and optimizing performance for concurrent users. Special attention was given to error handling, state management, and the maintainability of the codebase, allowing for future scalability and feature expansion.

The resulting application not only streamlines library workflows but also demonstrates best practices in web development, including modular design, responsive interfaces, and adherence to security standards. As web applications continue to evolve, projects like this highlight the importance of integrating user-centric design, scalable infrastructure, and robust security to deliver reliable digital services that meet the needs of modern institutions.

2.Goal

The main goal of this project is to develop a full-stack Library Management System that streamlines the management of books, users, borrowing, reservations, and fines for a library. The system aims to:

- Provide an intuitive web interface for both users and administrators.
- Enable efficient book catalog management, including adding, updating, and removing books.
- Allow users to search, borrow, reserve, and return books easily.
- Automate fine calculation and management for overdue books.
- Support user registration, authentication, and profile management.
- Notify users via email about important events (borrow confirmations, due reminders, overdue alerts, reservation readiness, and return confirmations).
- Offer administrative tools for monitoring library activity and managing users.
- Ensure data consistency, security, and a responsive user experience through a robust backend (Java Spring Boot) and a modern frontend (React.js).

3. Techniques & Tools Used

Backend (Java, Spring Boot)
- Spring Boot: For rapid backend development and RESTful API creation
- Spring Data JPA: For database access and ORM
- Maven: Project build and dependency management (pom.xml)
- JavaMail: For sending email notifications
- JUnit: For unit and integration testing
- application.properties: For environment and database configuration

Frontend (React.js)
- React.js: For building a dynamic, component-based user interface
- React Router: For client-side routing and navigation

- Context API: For global state management (authentication, user info)
- Axios/Fetch: For HTTP requests to backend APIs
- CSS Modules: For component-scoped styling

Database
- SQL: For schema definition
- Relational Database: MySQL

Development & Utilities
- Node.js & npm: For managing frontend dependencies and scripts
- Maven Wrapper: For consistent Maven builds across environments
- Shell/Batch Scripts: For starting backend and frontend servers (start.bat, start.sh)
- PowerShell: For API testing (test_api.ps1)
- IDE: Visual Studio Code,

II.REQUIREMENT ANALYSIS AND DESIGN

Requirement Analysis

Functional Requirements

Use Case Diagram

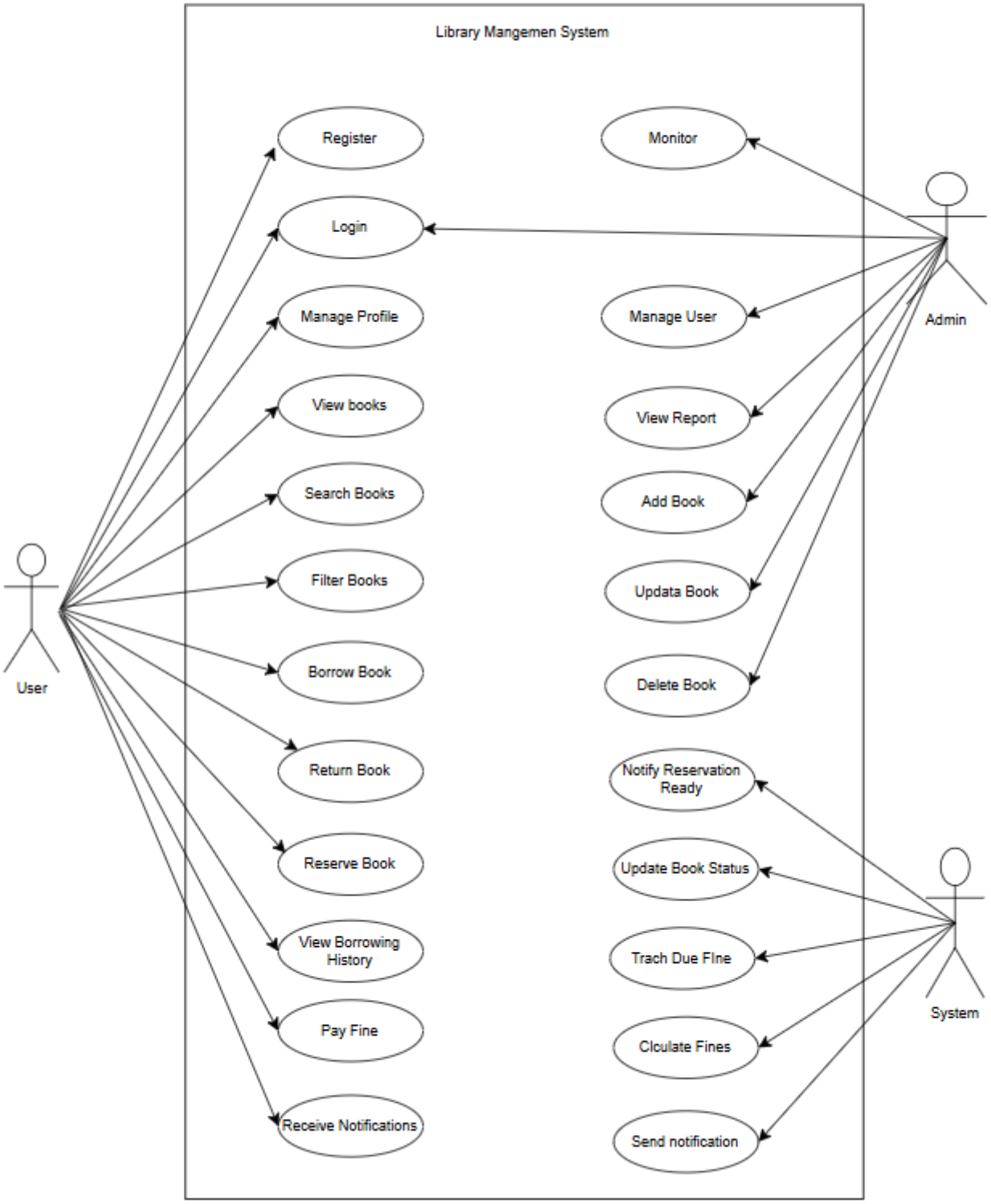

Figure 2 Use case

- **User**

Use Case: Register User Account

- Name: Register User Account
- Identifier: UC-User-Register
- Input:
  1. User provides registration details (name, email, password, etc.)
- Outputs:
  1. New user account created in the system
  2. Confirmation message to the user
- Basic Course

Actor: User, System	
1. User opens the registration page	1.1. System displays the registration form
2. User enters required information	2.1. System validates input
3. User submits the form	3.1. System creates a new user account
4. System confirms registration to the user	

- Preconditions:
  1. User is not already registered
- Post condition:
  1. User account is created and ready for login
- User Story:
  As a new user, I want to register an account so I can use the library system.

Use Case: Log In

- Name: Log In
- Identifier: UC-User-Login
- Input:
  1. User provides login credentials (email, password)
- Outputs:
  1. User authenticated and session started
  2. Access to user dashboard or home page
- Basic Course

Actor: User, System	
1. User opens the login page	1.1. System displays the login form
2. User enters credentials	2.1. System verifies credentials
3. If valid, user is logged in and redirected to dashboard	
4. If invalid, system displays error message	

- Preconditions:
  1. User is registered
- Post condition:
  1. User is authenticated and can access system features
- User Story:
  As a user, I want to log in so I can access my library account and services.

Use Case: Manage User Profile

- Name: Manage User Profile
- Identifier: UC-User-ManageProfile
- Input:
  1. User updates profile information (name, email, password, etc.)
- Outputs:
  1. User profile updated in the system
  2. Confirmation message to the user
- Basic Course

Actor: User, System	
1. User navigates to profile management page	1.1. System displays current profile information
2. User edits and submits changes	2.1. System validates and updates profile
3. System confirms update to the user	

- Preconditions:
  1. User is authenticated
- Post condition:
  1. User profile reflects the latest information
- User Story:
  As a user, I want to manage my profile so I can keep my account information up to date.

- Book Management

- **Admins**

Use Case: Add Book

- Name: Add Book
- Identifier: UC-Book-Add
- Input:
  1. Book details (title, author, ISBN, category, etc.)
- Outputs:
  1. New book record created in the database
  2. Confirmation message to admin
- Basic Course

Actor: Admin, System	
1. Admin opens the add book form	1.1. System displays the form
2. Admin enters book details	2.1. System validates input
3. Admin submits the form	3.1. System creates a new book record
4. System confirms addition to admin	

- Preconditions:
  1. Admin is authenticated
- Post condition:
  1. Book is available in the catalog

- User Story:
  As an admin, I want to add new books so users can borrow them.

Use Case: Update Book

- Name: Update Book
- Identifier: UC-Book-Update
- Input:
  1. Book identifier (ID)
  2. Updated book details
- Outputs:
  1. Book record updated in the database
  2. Confirmation message to admin
- Basic Course

Actor: Admin, System	
1. Admin selects a book to update	1.1. System displays current details
2. Admin edits book information	2.1. System validates input
3. Admin submits changes	3.1. System updates the book record
4. System confirms update to admin	

- Preconditions:
  1. Admin is authenticated
  2. Book exists in the system
- Post condition:
  1. Book details are updated in the catalog
- User Story:
  As an admin, I want to update book information to keep the catalog accurate.

Use Case: Delete Book

- Name: Delete Book
- Identifier: UC-Book-Delete
- Input:
  1. Book identifier (ID)
- Outputs:
  1. Book record removed from the database
  2. Confirmation message to admin
- Basic Course

Actor: Admin, System	
1. Admin selects a book to delete	1.1. System displays confirmation prompt
2. Admin confirms deletion	2.1. System removes the book record
3. System confirms deletion to admin	

- Preconditions:
  1. Admin is authenticated

2. Book exists in the system

- Post condition:
  1. Book is no longer available in the catalog
- User Story:
  As an admin, I want to delete books that are no longer available or needed.

- **Users**

Use Case: View Books

- Name: View Books
- Identifier: UC-Book-View
- Input:
  1. User navigates to the book list/catalog page
- Outputs:
  1. List of all available books displayed
- Basic Course

Actor: User, System	
1. User opens the book list page	1.1. System retrieves and displays all books

- Preconditions:
  1. User is authenticated (if required)
- Post condition:
  1. User can browse the entire book catalog
- User Story:
  As a user, I want to view all books so I can browse the library's collection.

Use Case: Search Books

- Name: Search Books
- Identifier: UC-Book-Search
- Input:
  1. Search keywords (e.g., title, author, ISBN)
- Outputs:
  1. List of books matching the search keywords
- Basic Course

Actor: User, System	
1. User enters search keywords in the search bar	1.1. System processes the keywords
2. User submits the search	2.1. System queries the database and displays matching books

- Preconditions:
  1. User is authenticated (if required)
- Post condition:
  1. User sees books relevant to their search
- User Story:
  As a user, I want to search for books by keywords so I can quickly find specific books.

Use Case: Filter Books

- Name: Filter Books
- Identifier: UC-Book-Filter
- Input:
  1. Filter criteria (e.g., category, author, availability)
- Outputs:
  1. List of books matching the selected filters
- Basic Course

Actor: User, System
1. User selects filter options from the filter menu 1.1. System applies filters
2. System displays books that match the selected criteria

- Preconditions:
  1. User is authenticated (if required)
- Post condition:
  1. User sees a refined list of books based on their preferences
- User Story:
  As a user, I want to filter books by category or availability so I can narrow down my choices.

- **Borrowing & Returning**

Use Case: Borrow Book

- Name: Borrow Book
- Identifier: UC-Book-Borrow
- Input:
  1. Selection of an available book by the user
- Outputs:
  1. Borrow record created in the database
  2. Book status updated to "borrowed"
  3. Confirmation message to the user
- Basic Course

Actor: User, System
1. User views the list of available books 1.1. System displays available books
2. User selects a book to borrow 2.1. System checks book availability
3. User confirms borrowing 3.1. System creates a borrow record and updates book status
4. System displays confirmation to the user

- Preconditions:
  1. User is authenticated
  2. Book is available for borrowing
- Post condition:
  1. Book is marked as borrowed and associated with the user
- User Story:

As a user, I want to borrow available books so I can read them and track my loans.

Use Case: Track Due Dates

- Name: Track Due Dates
- Identifier: UC-Book-TrackDueDate
- Input:
  1. Borrow record creation (book, user, borrow date)
- Outputs:
  1. Due date assigned and stored with the borrow record
  2. Overdue status flagged if book is not returned by due date
- Basic Course

Actor: System	
1. User borrows a book	1.1. System calculates and assigns a due date
2. System monitors due dates	2.1. System flags overdue status if due date is passed

- Preconditions:
  1. User is authenticated
  2. Book is borrowed through the system
- Post condition:
  1. Each borrow record has an accurate due date and overdue status if applicable
- User Story:

As a user, I want the system to track due dates so I know when to return borrowed books.

Use Case: Track Borrowing History

- Name: Track Borrowing History
- Identifier: UC-Book-TrackHistory
- Input:
  1. Borrow and return events (book, user, dates)
- Outputs:
  1. Borrowing history maintained for each user
  2. Users can view their past and current borrow records
- Basic Course

Actor: User, System	
1. User borrows or returns a book	1.1. System records the event in the user's borrowing history
2. User views borrowing history	2.1. System displays all borrow and return records for the user

- Preconditions:
  1. User is authenticated
- Post condition:
  1. User's borrowing history is up-to-date and accessible
- User Story:

As a user, I want to view my borrowing history so I can keep track of the books I have borrowed and returned.

Users can return books; system updates status.

Use Case: Return Book

- Name: Return Book
- Identifier: UC-Book-Return
- Input:
  1. User selects a borrowed book to return
- Outputs:
  1. Return event recorded in the system
  2. Confirmation message to the user
- Basic Course

Actor: User, System	
1. User views their list of borrowed books	1.1. System displays borrowed books
2. User selects a book to return	2.1. System prompts for confirmation
3. User confirms return	3.1. System records the return event
4. System displays confirmation to the user	

- Preconditions:
  1. User is authenticated
  2. Book is currently borrowed by the user
- Post condition:
  1. Return event is recorded for the user and book
- User Story:
  As a user, I want to return borrowed books so I can complete my loan and avoid fines.

Use Case: Update Book Status After Return

- Name: Update Book Status After Return
- Identifier: UC-Book-UpdateStatus
- Input:
  1. Book return event
- Outputs:
  1. Book status updated to "available" in the system
  2. Borrow record updated with return date
- Basic Course

Actor: System
1. System detects a book return event
2. System updates the book’s status to "available"
3. System updates the borrow record with the return date

- Preconditions:
  1. Book is returned through the system
- Post condition:

1. Book is available for borrowing by others
2. Borrow record reflects the return

- User Story:

As a user, I want the system to update the book's status after I return it so others can borrow it.

- Reservation System

- **Users can reserve books that are currently borrowed.**

Use Case: Reserve Borrowed Book

- Name: Reserve Borrowed Book
- Identifier: UC-Book-Reserve
- Input:
  1. User selects a book that is currently borrowed by someone else
- Outputs:
  1. Reservation record created in the system
  2. Confirmation message to the user
- Basic Course

Actor: User, System
1. User views a book that is currently borrowed 1.1. System displays book status as "borrowed"
2. User clicks the "Reserve" button 2.1. System prompts for confirmation
3. User confirms reservation 3.1. System creates a reservation record for the user and book
4. System displays reservation confirmation to the user

- Preconditions:
  1. User is authenticated
  2. Book is currently borrowed (not available)
- Post condition:
  1. Reservation is recorded and user is in the reservation queue
- User Story:

As a user, I want to reserve books that are currently borrowed so I can borrow them when they become available.

- **System notifies users when reserved books become available.**

Use Case: Notify User When Reserved Book Becomes Available

- Name: Notify User of Available Reserved Book
- Identifier: UC-Book-NotifyReservation
- Input:
  1. Book return event for a book with active reservations
- Outputs:
  1. Notification (e.g., email) sent to the next user in the reservation queue
  2. Reservation status updated
- Basic Course

Actor: System, User

1. System detects a book has been returned 1.1. System checks for active reservations on the book
2. System identifies the next user in the reservation queue
3. System sends a notification (email/message) to the user 3.1. System updates reservation status to "ready for pickup"
4. User receives notification and can borrow the book

- Preconditions:
  1. Book has at least one active reservation
  2. User has a valid contact method (e.g., email)
- Post condition:
  1. User is informed that the reserved book is available for borrowing
- User Story:
  As a user, I want to be notified when a reserved book becomes available so I can borrow it promptly.

- **Fine Management**
- **System calculates fines for overdue books.**

Use Case: Calculate Fines for Overdue Books

- Name: Calculate Fines for Overdue Books
- Identifier: UC-Fine-Calculate
- Input:
  1. Borrow record with due date and actual return date
- Outputs:
  1. Fine amount calculated and recorded if the book is returned late
  2. Fine details associated with the user and borrow record
- Basic Course

Actor: System
1. System detects a book is returned 1.1. System compares actual return date with due date
2. If the book is overdue, system calculates the fine based on the number of overdue days and the fine rate
3. System creates or updates a fine record for the user and borrow record
4. System notifies the user of the fine

- Preconditions:
  1. Book is returned after the due date
  2. Fine policy/rate is defined in the system
- Post condition:
  1. Fine is recorded and can be viewed/managed by user and admin
- User Story:
  As a user, I want the system to calculate fines for overdue books so I am aware of any penalties for late returns.

- **Users and admins can view and manage fines.**

Use Case: Calculate Fines for Overdue Books

- Name: Calculate Fines for Overdue Books
- Identifier: UC-Fine-Calculate
- Input:
  1. Borrow record with due date and actual return date
- Outputs:
  1. Fine amount calculated and recorded if the book is returned late
  2. Fine details associated with the user and borrow record
- Basic Course

Actor: System
1. System detects a book is returned 1.1. System compares actual return date with due date
2. If the book is overdue, system calculates the fine based on the number of overdue days and the fine rate
3. System creates or updates a fine record for the user and borrow record
4. System notifies the user of the fine

- Preconditions:
  1. Book is returned after the due date
  2. Fine policy/rate is defined in the system
- Post condition:
  1. Fine is recorded and can be viewed/managed by user and admin
- User Story:
  As a user, I want the system to calculate fines for overdue books so I am aware of any penalties for late returns.

- **Email Notifications: Automated emails for borrow confirmations, due reminders, overdue alerts, reservation readiness, and return confirmations.**

Use Case: Send Automated Email Notifications

- Name: Send Automated Email Notifications
- Identifier: UC-Notification-AutomatedEmail
- Input:
  1. Library events (borrow, due date approaching, overdue, reservation ready, book returned)
  2. User's email address
- Outputs:
  1. Email sent to the user with relevant information about the event
  2. Notification record (optional, for audit/logging)
- Basic Course

Actor: System, User
1. System detects a relevant event (e.g., book borrowed, due soon, overdue, reservation ready, book returned)
2. System generates an email message with event details

3. System sends the email to the user's registered email address
4. User receives the email notification

- Preconditions:
  1. User has a valid email address in the system
  2. Event occurs that triggers a notification
- Post condition:
  1. User is informed about important library events via email
- User Story:
  As a user, I want to receive automated email notifications about my library activities so I stay informed about my loans, reservations, and returns.

- **Admin Dashboard:Admins can monitor library activity, manage users, and view reports.**

Use Case: Monitor Library Activity

- Name: Monitor Library Activity
- Identifier: UC-Admin-MonitorActivity
- Input:
  1. Admin accesses the dashboard or activity logs
- Outputs:
  1. Display of current and recent library activities (borrowing, returns, reservations, fines, etc.)
- Basic Course

Actor: Admin, System
1. Admin logs in and navigates to the activity monitoring section 1.1. System retrieves and displays activity data

- Preconditions:
  1. Admin is authenticated
- Post condition:
  1. Admin can view up-to-date library activity
- User Story:
  As an admin, I want to monitor library activity so I can oversee operations and respond to issues.

Use Case: Manage Users

- Name: Manage Users
- Identifier: UC-Admin-ManageUsers
- Input:
  1. User management actions (add, update, delete, view user details, reset password, change roles)
- Outputs:
  1. User records created, updated, or deleted in the system
  2. Confirmation messages for each action

- Basic Course

Actor: Admin, System	
1. Admin navigates to the user management section	1.1. System displays user list
2. Admin selects an action (add, edit, delete, etc.)	2.1. System processes the action and updates user records
3. System confirms the action to the admin	

- Preconditions:
  1. Admin is authenticated
- Post condition:
  1. User records are updated as per admin actions
- User Story:
  As an admin, I want to manage users so I can maintain accurate and secure user accounts.

Use Case: View Reports

- Name: View Reports
- Identifier: UC-Admin-ViewReports
- Input:
  1. Admin selects report type and parameters (e.g., borrowing statistics, overdue books, fines collected)
- Outputs:
  1. Report generated and displayed to the admin
- Basic Course

Actor: Admin, System	
1. Admin navigates to the reports section	1.1. System displays available report options
2. Admin selects a report and parameters	2.1. System generates and displays the report

- Preconditions:
  1. Admin is authenticated
- Post condition:
  1. Admin can view and analyze library data through reports
- User Story:
  As an admin, I want to view reports so I can analyze library performance and make informed decisions.

2. Non-Functional Requirements

1. Operational Requirements

- All components of the library management system must operate reliably and efficiently.
- The hosting environment must be properly licensed and maintained to handle all user and admin requests.

- An administrator should regularly update the software, monitor system health, and address any detected errors or vulnerabilities.

2. Legal Requirements

- All third-party libraries, frameworks, and resources integrated into the system must be properly cited to avoid copyright infringement.
- The project's original source code must be kept confidential and protected from unauthorized access or distribution.

3. Usability Requirements

- The system should be easy to learn and use for all user types, including students, staff, and administrators.
- Comprehensive user manuals and help documentation must be provided.
- Error messages should be clear, informative, and provide hints or links to solutions.
- The user interface should be well-designed, intuitive, and consistent across all devices.
- The system should support efficient workflows, allowing users to accomplish tasks with minimal errors and steps.

4. Humanity Requirements

- The system should be accessible to users with little or no technical background.
- Detailed documentation and tooltips should be available to help users understand system features.
- The graphical user interface must be attractive, responsive, and require minimal time for users to become proficient.
- Accessibility standards (such as contrast, font size, and keyboard navigation) should be followed to support all users.

5. Performance Requirements

a. Response Requirements:

- Login, search, and book management actions must respond within 2 seconds under normal load.
- Database updates (e.g., borrowing, returning, reserving books) should be processed seamlessly and quickly.
- The system should support at least 10 concurrent transactions without noticeable delay.
- Web pages should load rapidly, displaying lists of 50+ books per page efficiently.

b. Throughput Requirements:

- The system must handle at least 1000 simultaneous users browsing the catalog without performance degradation.
- Multiple database operations (e.g., batch updates) should be processed efficiently.

c. Availability Requirements:

- The system should utilize effective memory management and garbage collection.
- The application should be available 99.5% of the time, with scheduled maintenance communicated in advance.

6. Maintainability Requirements

- The codebase should be modular, well-documented, and include meaningful comments to facilitate future updates and bug fixes.
- The database schema should be clear and stable, supporting easy maintenance and expansion.
- Admin interfaces should allow for easy management of books, users, and system settings.

7. Support Requirements

- Library staff and users should have access to support channels (e.g., helpdesk, email, or hotline) for reporting issues or requesting assistance.
- Remote support tools may be provided for direct troubleshooting by the development team.
- Regular system health checks and maintenance should be scheduled (e.g., monthly or quarterly).

8. Security Requirements

- All user data must be protected from unauthorized access using secure authentication and authorization mechanisms.
- Sensitive information (e.g., passwords, personal data) must be encrypted in transit and at rest.
- The system should restrict login attempts (e.g., lock account after five failed attempts for 24 hours).
- Only administrators may perform critical data modifications; all such actions must be logged with user, timestamp, and action details.
- All communications between clients and the server must be encrypted (e.g., HTTPS/SSL).
- User sessions should have timeouts and be securely managed.
- Regular backups of all system data must be performed, with copies stored securely offsite.
- A privacy policy must be in place to protect user information from third-party access.

9. Interface Requirements

- The system must support standard web protocols (HTTP/HTTPS) and RESTful API communication.
- Buttons and controls should use clear icons and labels matching their functions.
- Reports (e.g., borrowing statistics, overdue books) should be generated and available to admins.
- The interface should support multiple languages for diverse user groups.
- Error and buffer management should be robust, ensuring smooth operation even under heavy load.

3. System Requirements

- Backend: Java 21, Spring Boot, Maven
- Frontend: React.js, Node.js, npm
- Database: Relational DB (e.g., MySQL, PostgreSQL)

- Email Service: SMTP server for notifications
- Development Tools: IDE (VS Code, IntelliJ), Git

2.DESIGN
System Architecture Model

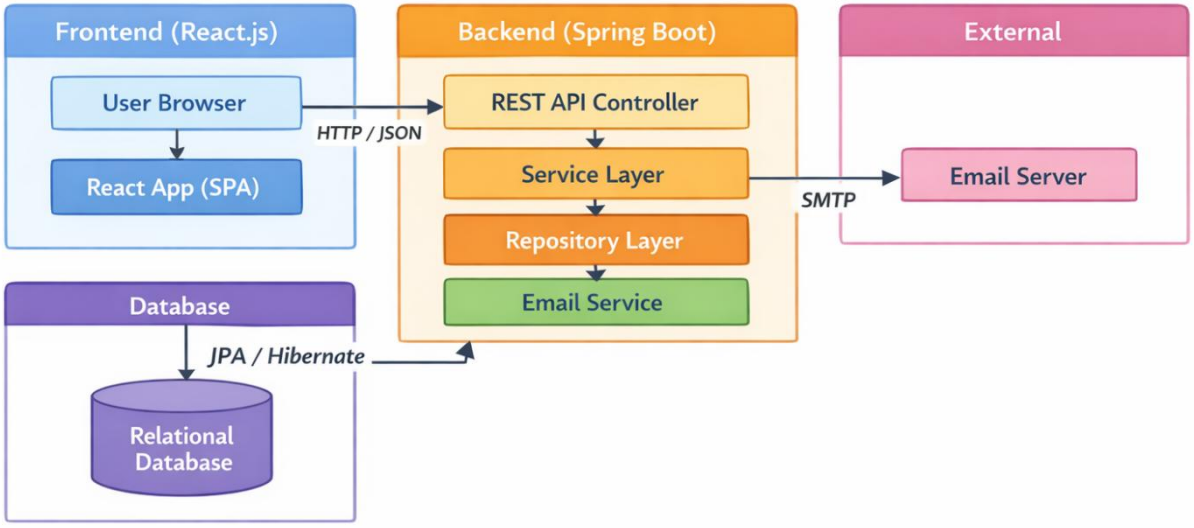

Figure 3 System Architecture Model

Entity-Relationship Diagram (ERD)

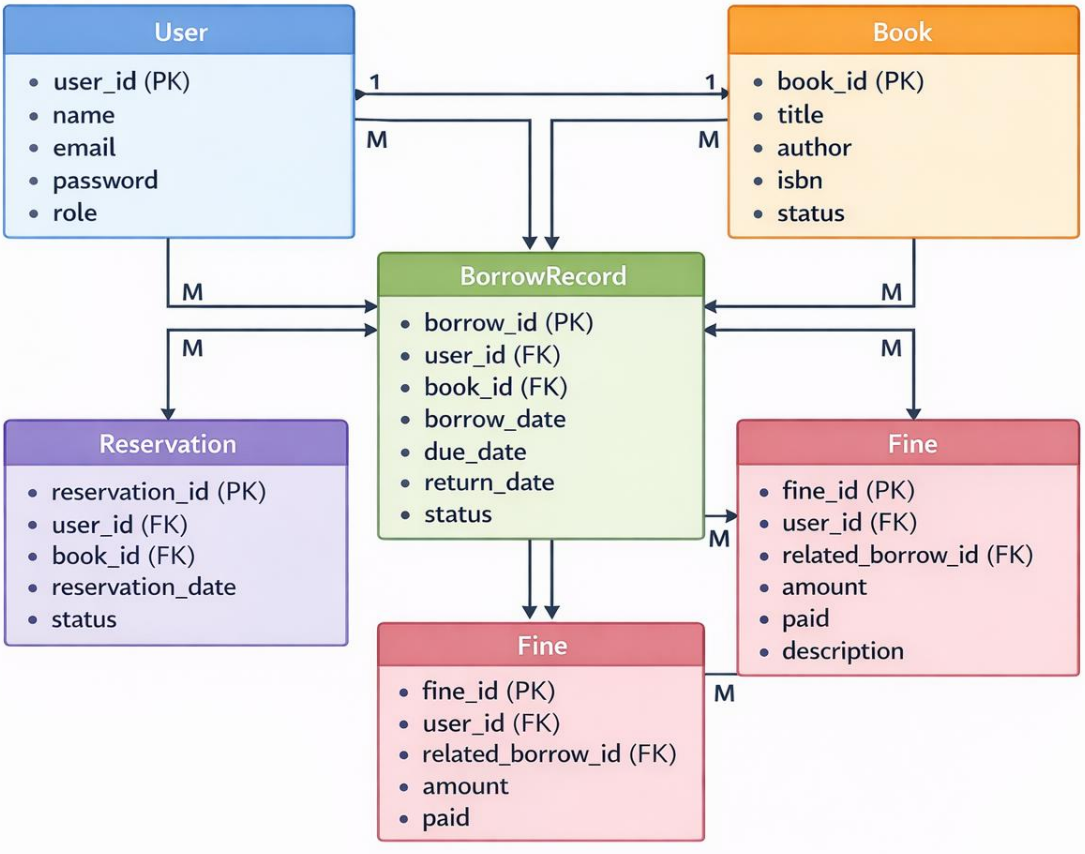

Figure 4 ERD

User

- user_id (PK, INT, AUTO_INCREMENT)
- name (VARCHAR)
- email (VARCHAR, UNIQUE)
- password (VARCHAR)
- role (VARCHAR) -- e.g., 'user', 'admin'
- created_at (DATETIME)
- updated_at (DATETIME)

Book

- book_id (PK, INT, AUTO_INCREMENT)
- title (VARCHAR)
- author (VARCHAR)
- isbn (VARCHAR, UNIQUE)
- category (VARCHAR)
- status (VARCHAR) -- e.g., 'available', 'borrowed', 'reserved'
- published_year (INT)
- created_at (DATETIME)
- updated_at (DATETIME)

BorrowRecord

- borrow_id (PK, INT, AUTO_INCREMENT)
- user_id (FK, INT) REFERENCES User(user_id)
- book_id (FK, INT) REFERENCES Book(book_id)
- borrow_date (DATE)
- due_date (DATE)
- return_date (DATE)
- status (VARCHAR) -- e.g., 'borrowed', 'returned', 'overdue'

Reservation

- reservation_id (PK, INT, AUTO_INCREMENT)
- user_id (FK, INT) REFERENCES User(user_id)
- book_id (FK, INT) REFERENCES Book(book_id)
- reservation_date (DATE)
- status (VARCHAR) -- e.g., 'active', 'fulfilled', 'cancelled'

Fine

- fine_id (PK, INT, AUTO_INCREMENT)
- user_id (FK, INT) REFERENCES User(user_id)
- borrow_id (FK, INT) REFERENCES BorrowRecord(borrow_id)
- amount (DECIMAL)
- paid (BOOLEAN)
- description (VARCHAR)
- issued_date (DATE)

AuditLog

- log_id (PK, INT, AUTO_INCREMENT)
- user_id (FK, INT) REFERENCES User(user_id)
- action (VARCHAR)

- object_type (VARCHAR)
- object_id (INT)
- prior_value (TEXT)
- new_value (TEXT)
- timestamp (DATETIME)

Class Diagram

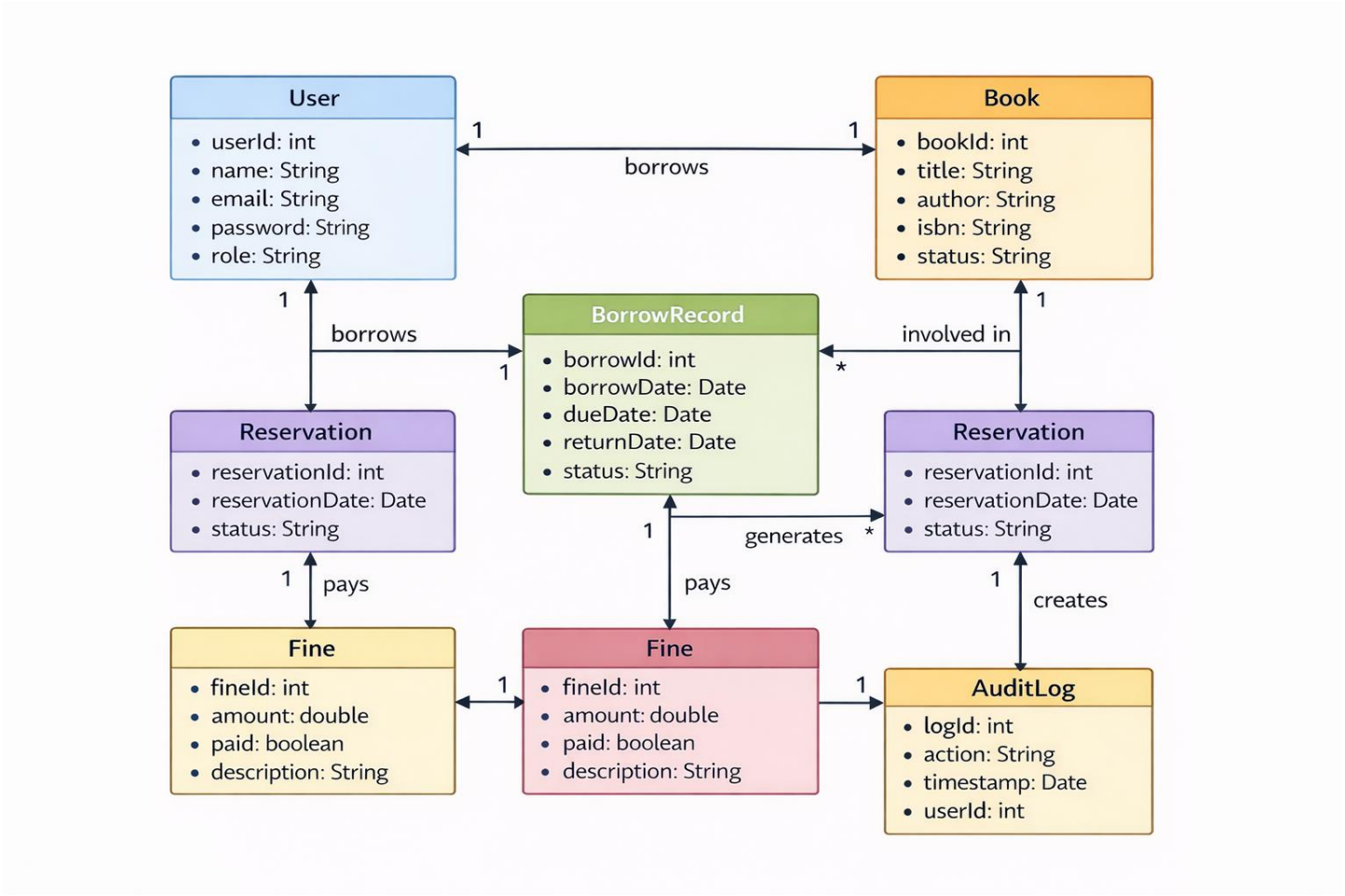

Figure 5 Class Diagram

Use Case Diagram

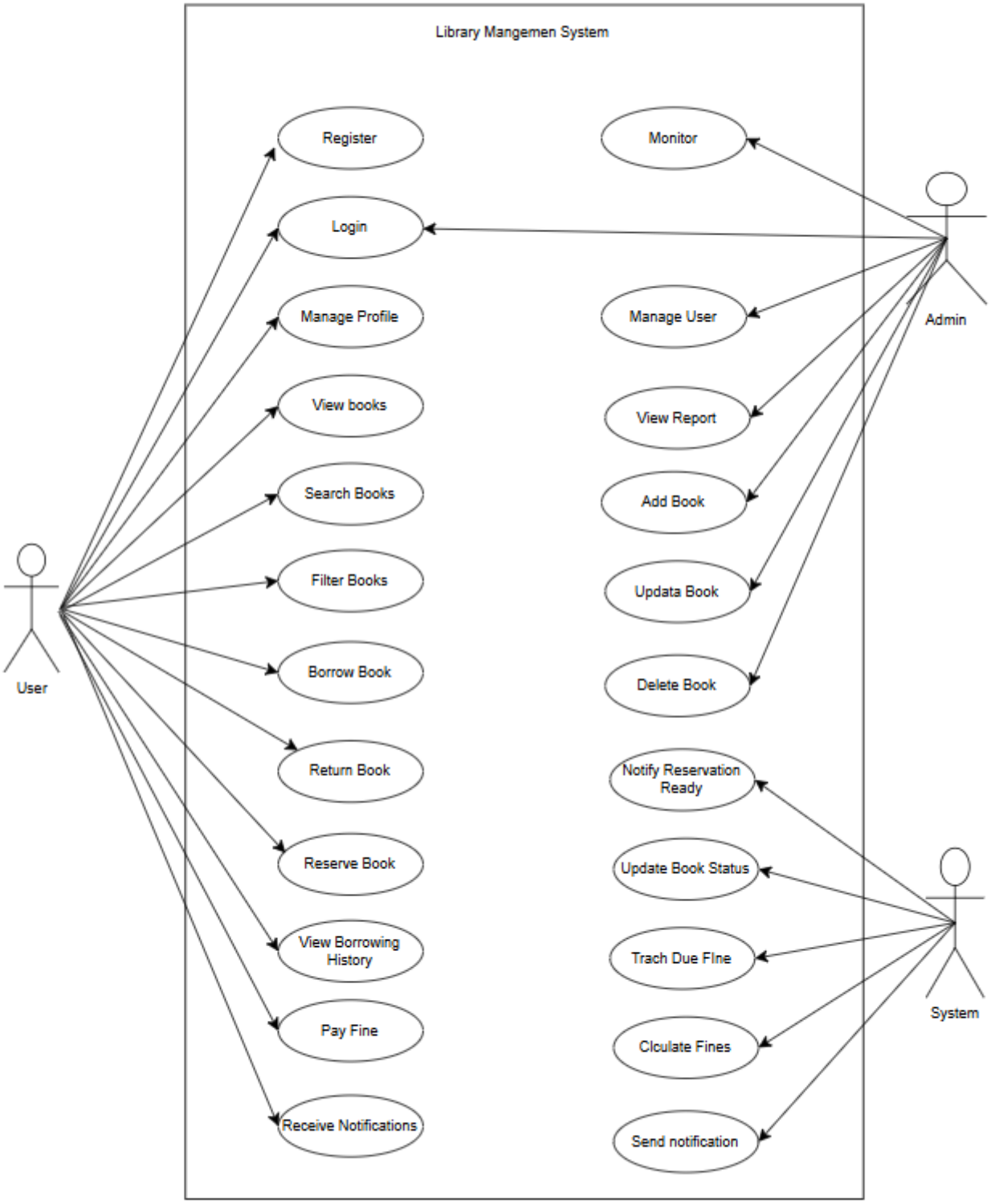

Figure 6 Use case Diagram

DEMO – RESULT

i.User functions

1. User's register

Step 1: On the Login Page, Click the Register Button

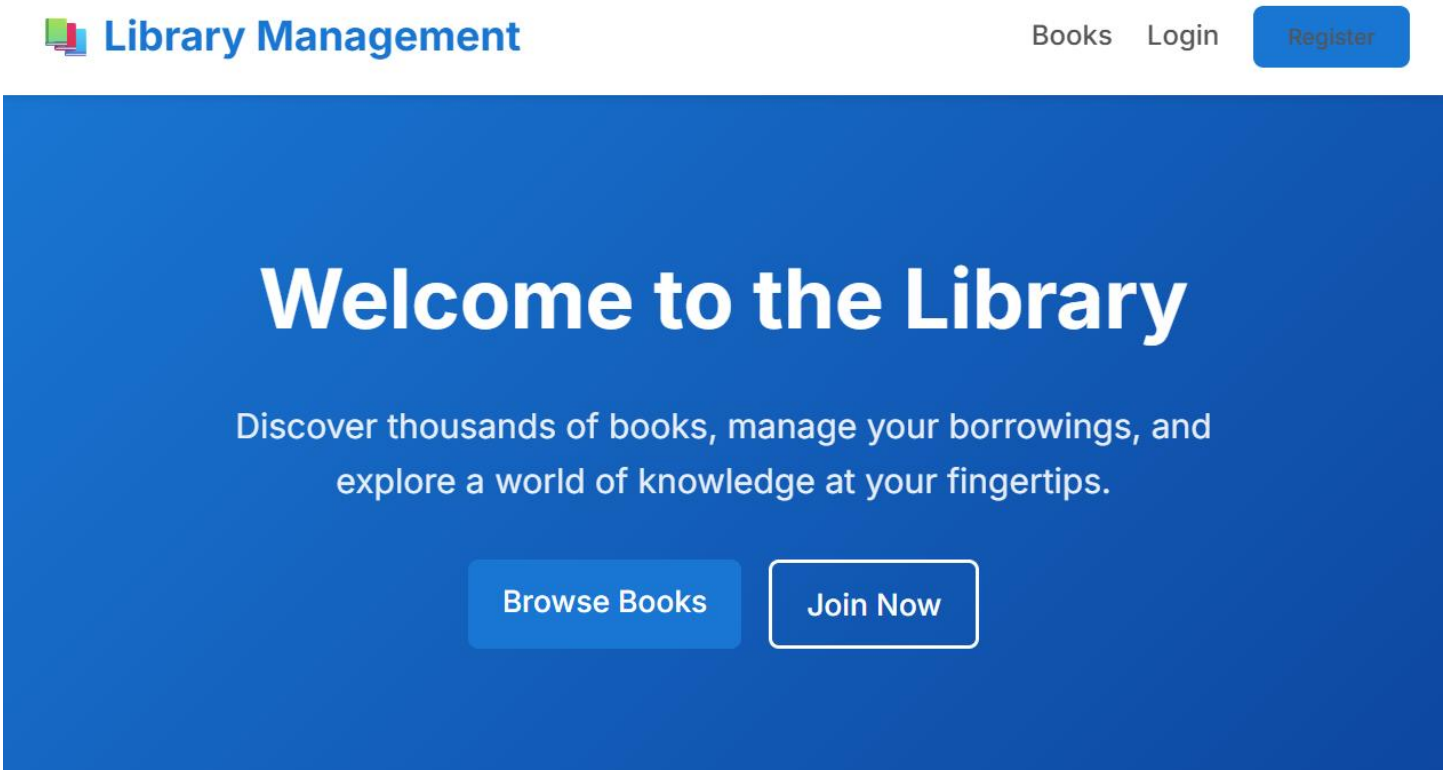

Figure 7 Main Page

- The registration page will appear, prompting the user to enter their account information.

Step 2: Fill in All Required Fields and Click Create Account to Complete Registration

- The complete registration form includes:
  - Full Name (*)
  - Email Address (*)
  - Password (*)
  - Confirm Password (*)
  - Phone Number

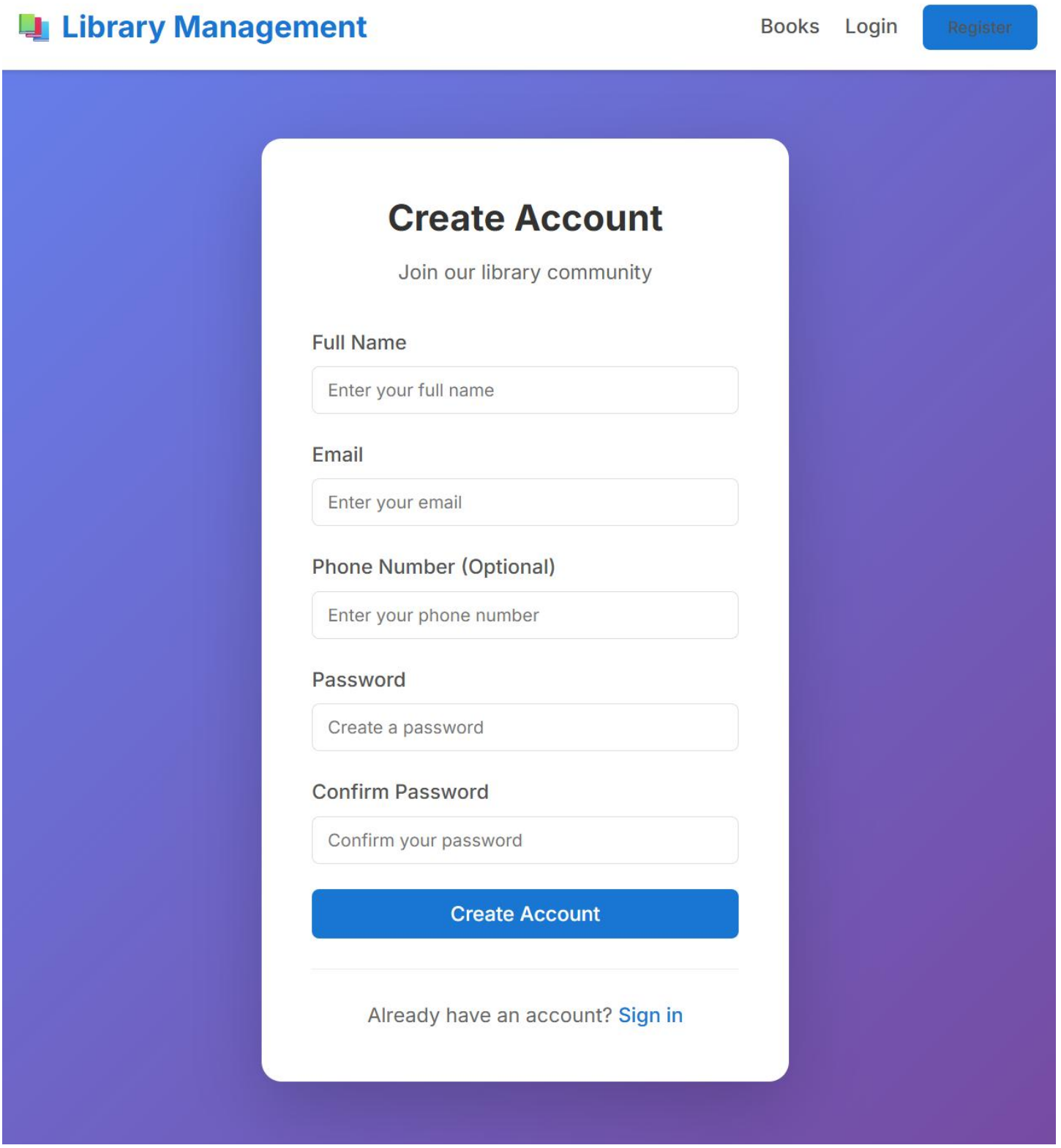

Figure 8 Register

2. User's Login

Step 1: On the Login Page, enter Your Login Credentials

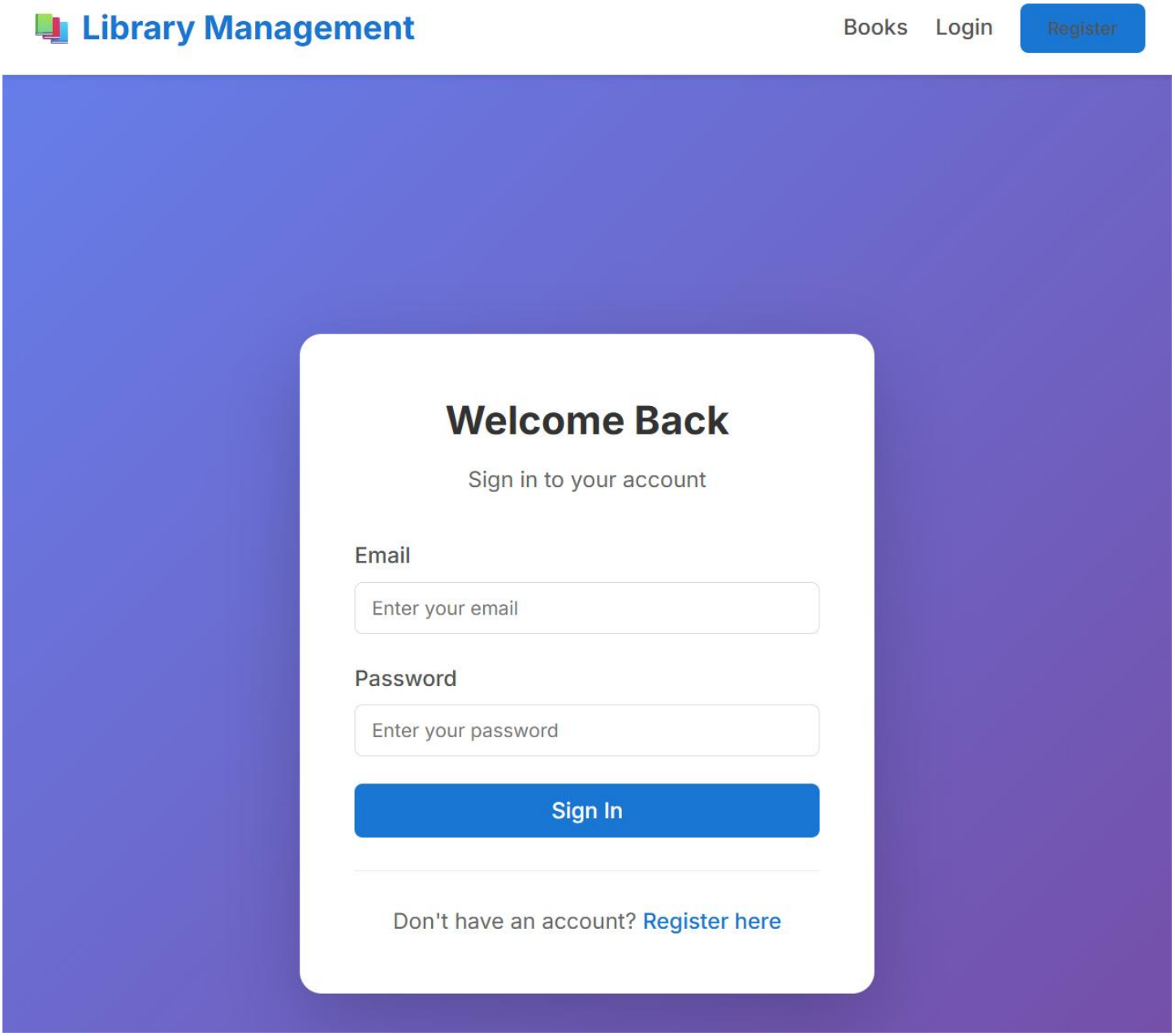

Figure 9 Log In

- *On the login page, fill in the following required fields:*
  - *Email Address (required)*
  - *Password (required)*

Step 3: Click the "Login" Button to Sign In

- *After entering your credentials, click the "Login" button to proceed.*

Step 4: Login Result

- *If your email and password are correct, you will be successfully logged in and*

redirected to your dashboard or home page.

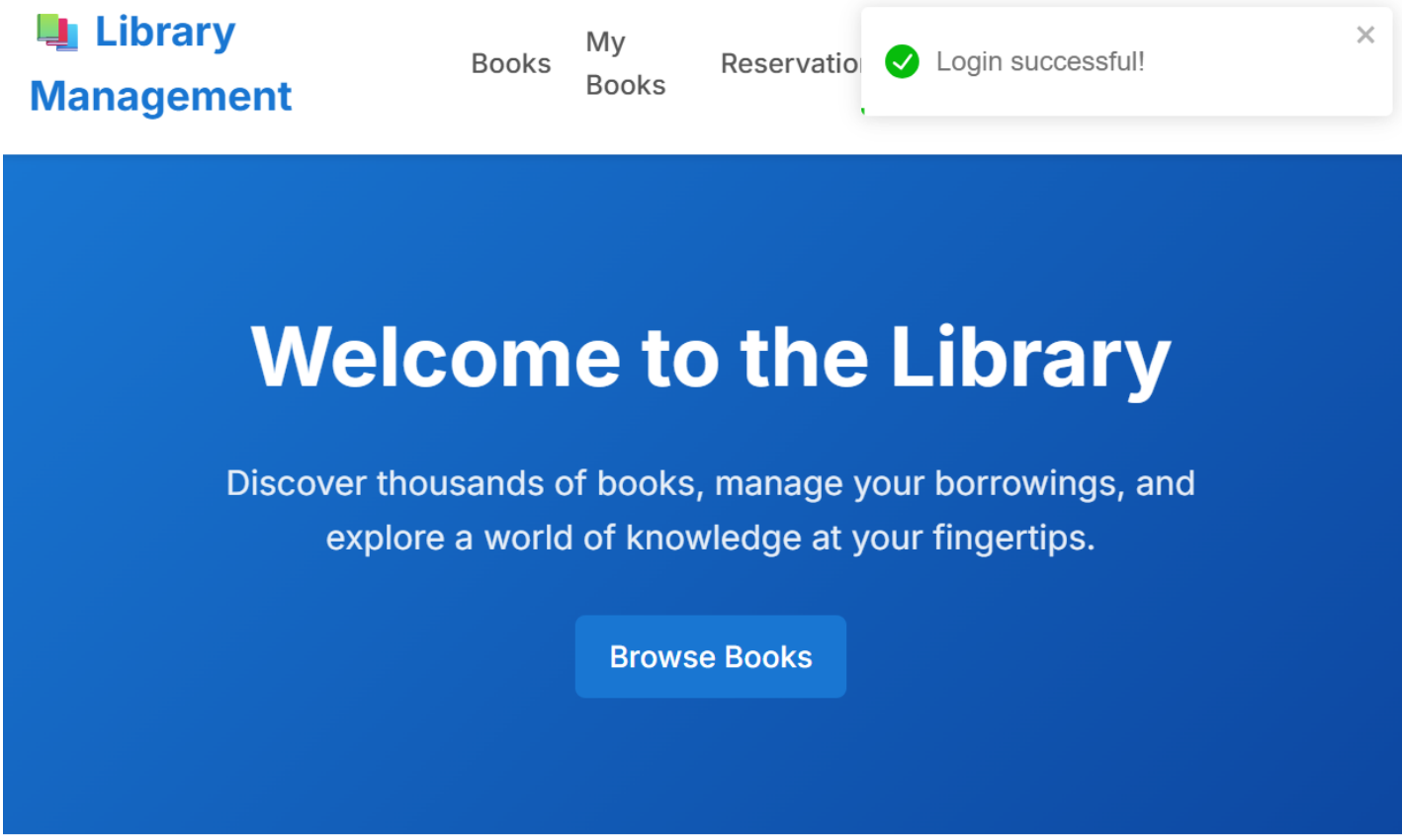

Figure 10 Result

- *If your credentials are incorrect, an error message will appear. Please check your email and password and try again.*

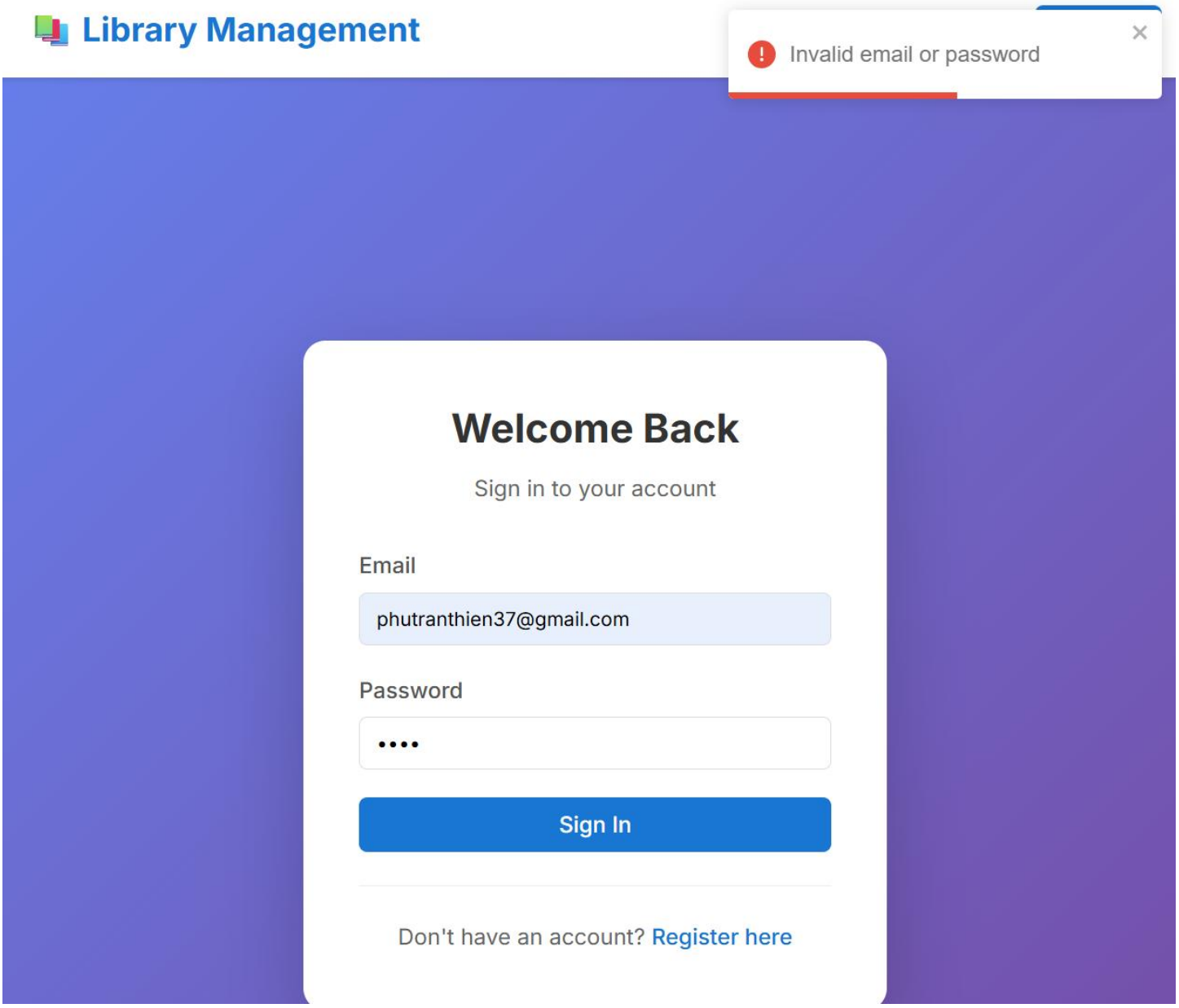

Figure 11 ERROR

3. Display user's profile

Step 1: Log In to Your Account

- *Enter your email and password on the login page.*
- *Click the "Login" button to access your account.*

Step 2: Navigate to the Profile Page

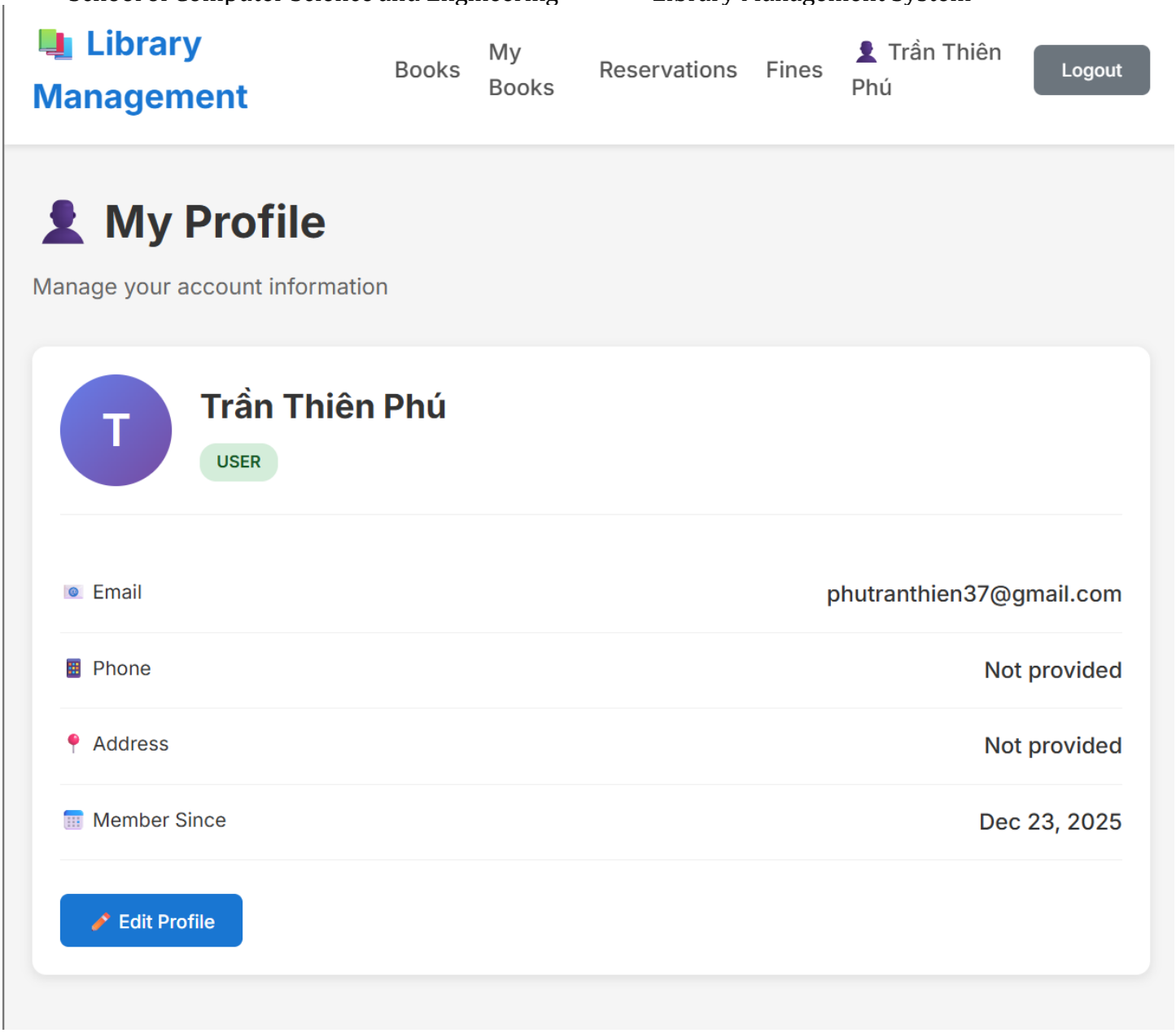

Figure 12 User Profile

- *Once logged in, locate the "User's name" option in the navigation*
- *Click on that to open your profile page.*

Step 3: View Your Profile Information

- *Your profile page will display your personal information, including:*
  - *Full Name*
  - *Email Address*
  - *Account creation date*
  - *Any other relevant details*
  - *Member Since*

Step 4: Edit Profile (Optional)

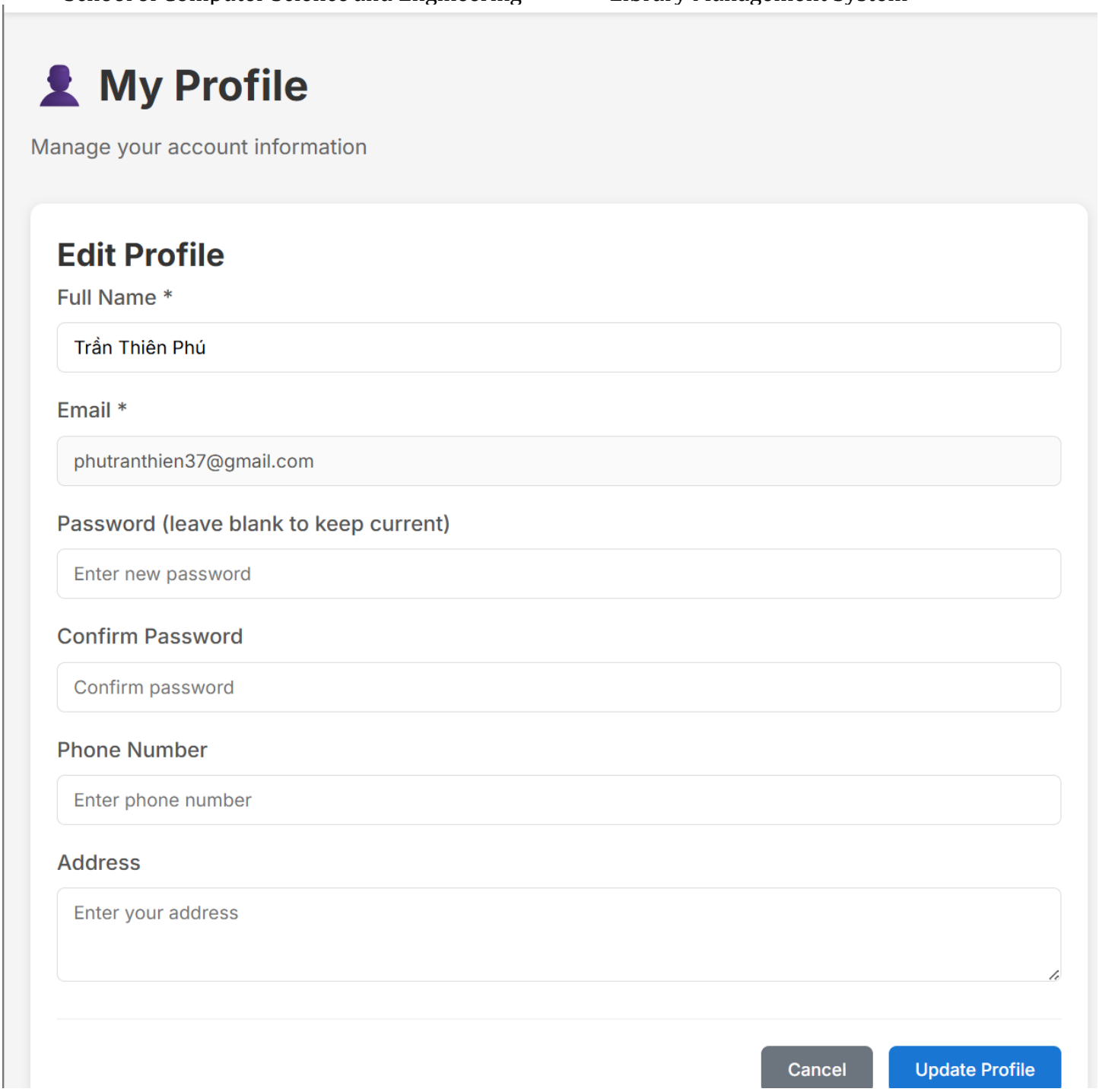

Figure 13 Edit User Profile

- *If you wish to update your information, click the "Edit Profile" button.*
- *Make the necessary changes and save to update your profile.*

II.LIBARY VIEWING FUNCTIONS

2.1. View a filtered product

Step 1: Log In to Your Account

- *Enter your email and password on the login page.*

- *Click the "Login" button to access your account.*

Step 2: Navigate to the Books Section

- *Click on it to open the main book listing page.*

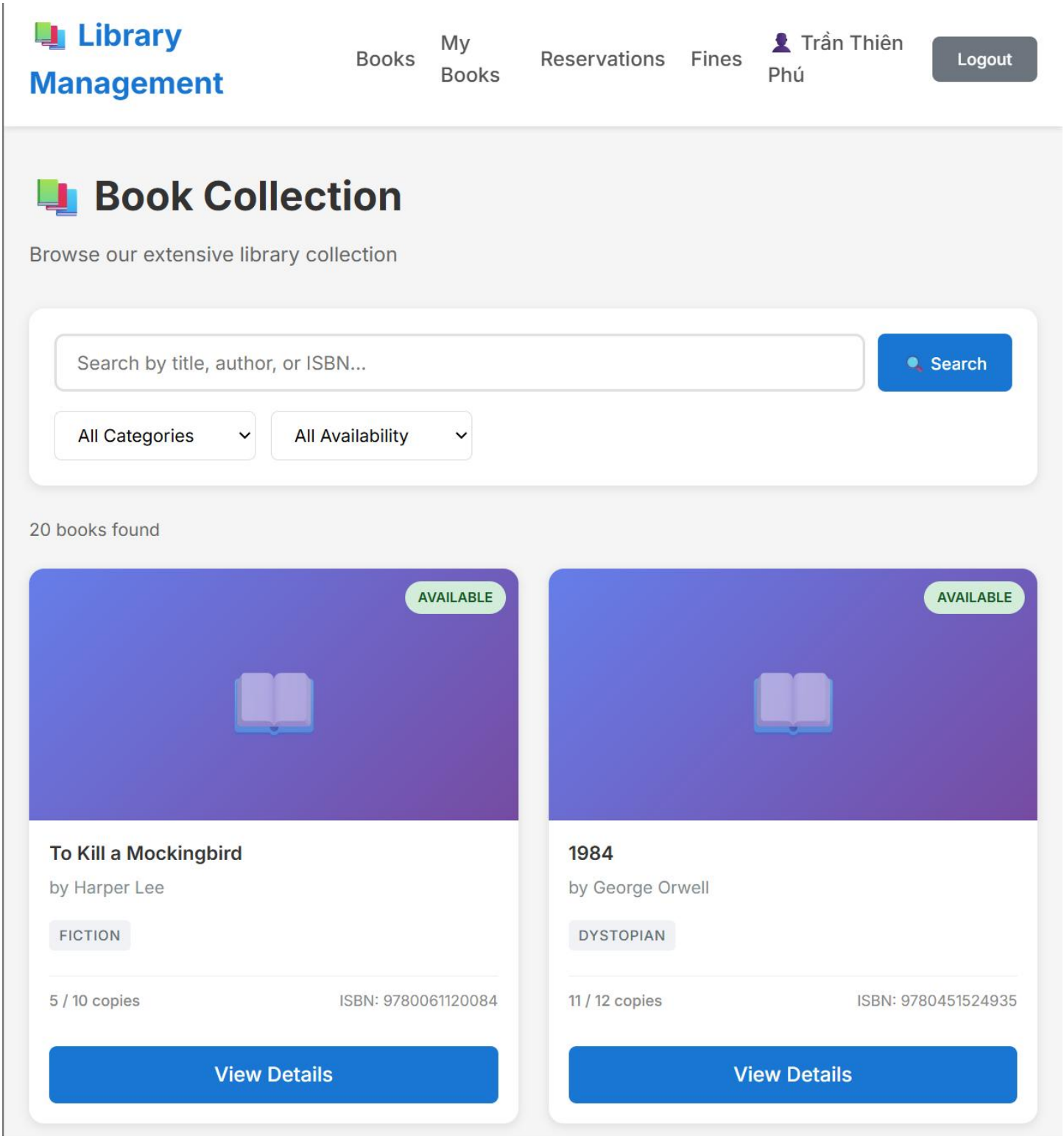

Figure 14 View Product

Step 3: Select a Category

- *On the catalog page, you will see a list of categories (e.g., Fiction, Non-Fiction, Science,*

History).

- *Click on the desired category to filter the products.*

Step 4: View Products in the Selected Category

- *The system will display all books belonging to the selected category.*
- *You can scroll through the list, view book details, and check availability.*

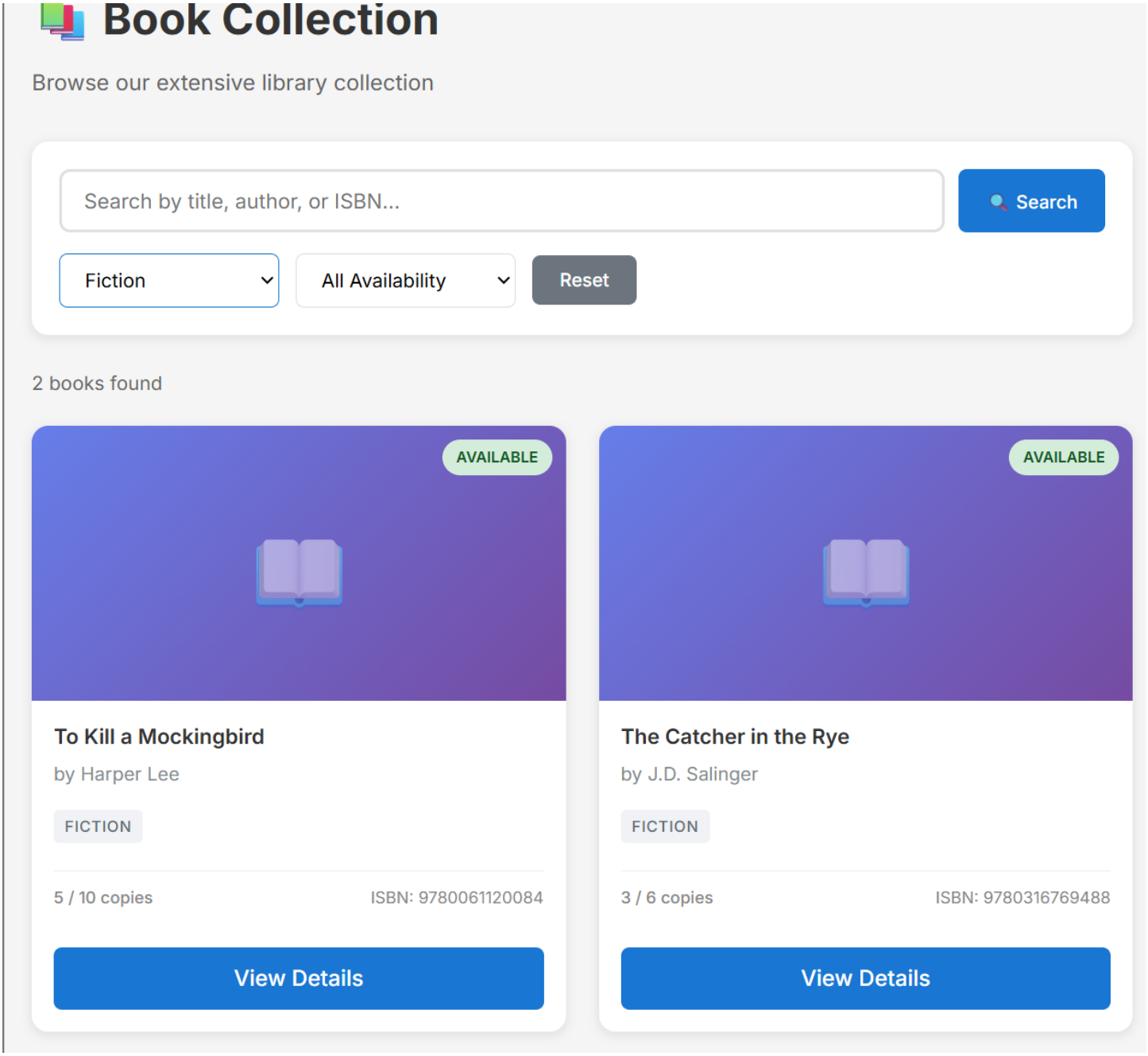

Figure 15 Filtered Search

2.2. View a specified product

Step 1: Log In to Your Account

- *Enter your email and password on the login page.*

- *Click the "Login" button to access your account.*

Step 2: Navigate to the Catalog or Books Section

- *Locate the "Catalog," "Books," or "Browse" option in the navigation menu.*
- *Click on it to open the main book listing page.*

Step 3: Search or Select the Desired Product

- *Use the search bar to enter the title, author, or ISBN of the book you want to view.*
- *Alternatively, browse through the list or category to find the specific book.*

Step 4: Click on the Specified Product

- *Once you locate the desired book, click on its title or cover image.*

Step 5: View Product Details

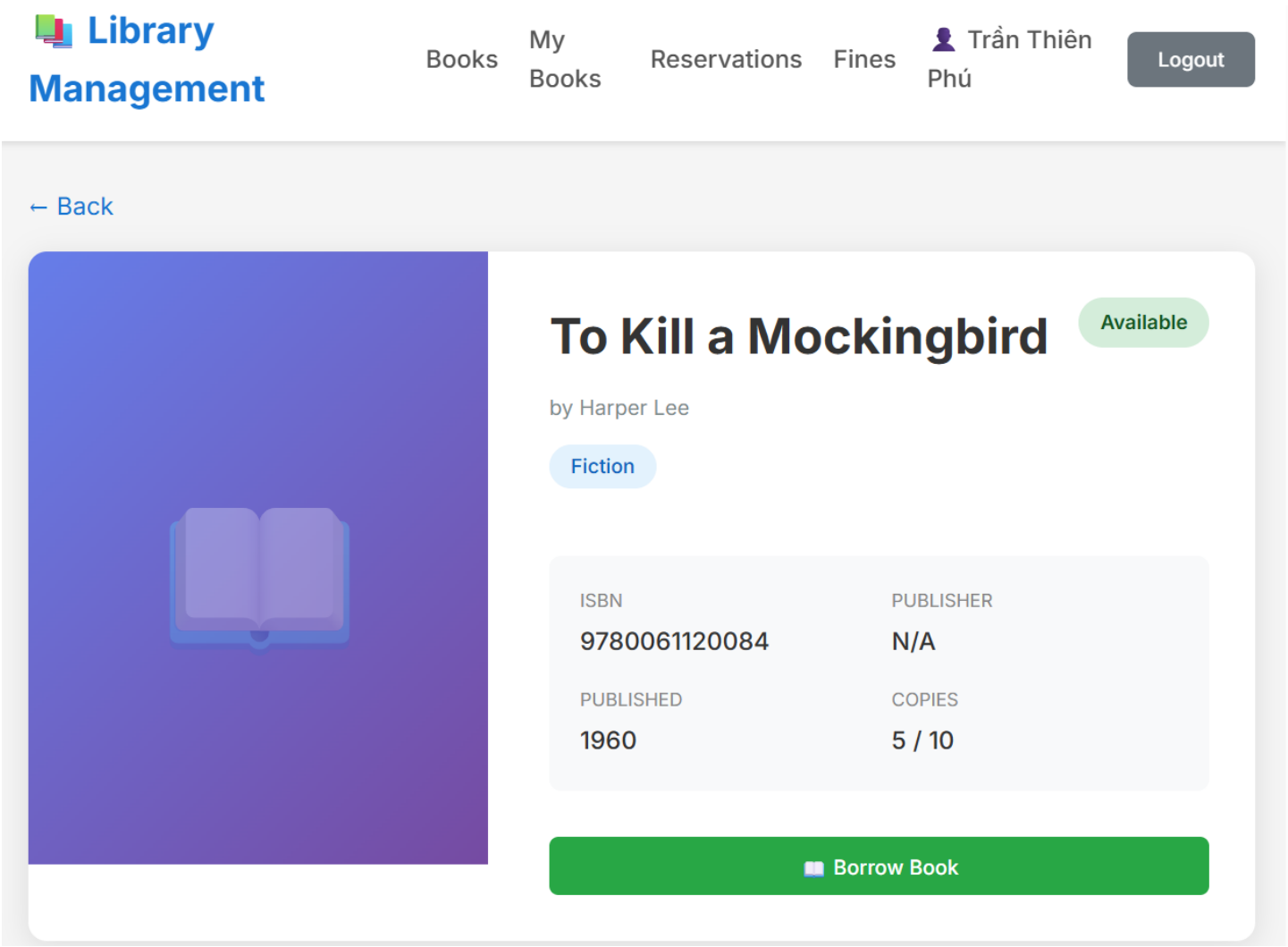

Figure 16 Sepecific search

- *The system will display the book's detail page, including:*
  - *Title*
  - *Author*

- *Category*
- *Description*
- *Availability status*
- *Publication year*
- *ISBN*
- *Cover image*

2.3. Search product by name/function

Step 1: Log In to Your Account

- *Enter your email and password on the login page.*
- *Click the "Login" button to access your account.*

Step 2: Navigate to the Catalog or Search Section

- *Locate the "Catalog," "Books," or "Search" option in the navigation menu.*
- *Click on it to open the search interface.*

Step 3: Enter Search Criteria

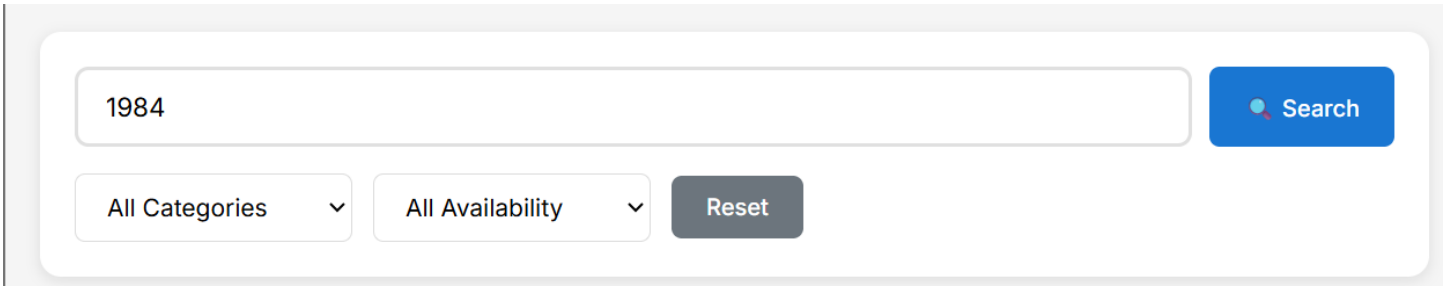

Figure 17 Search

- *In the search bar, type the name (title), author, ISBN, or relevant keyword of the book you want to find.*
- *You may also use advanced search options to filter by category, publication year, or other attributes.*

Step 4: Submit the Search

- *Click the "Search" button or press Enter to execute the search.*

Step 5: View Search Results

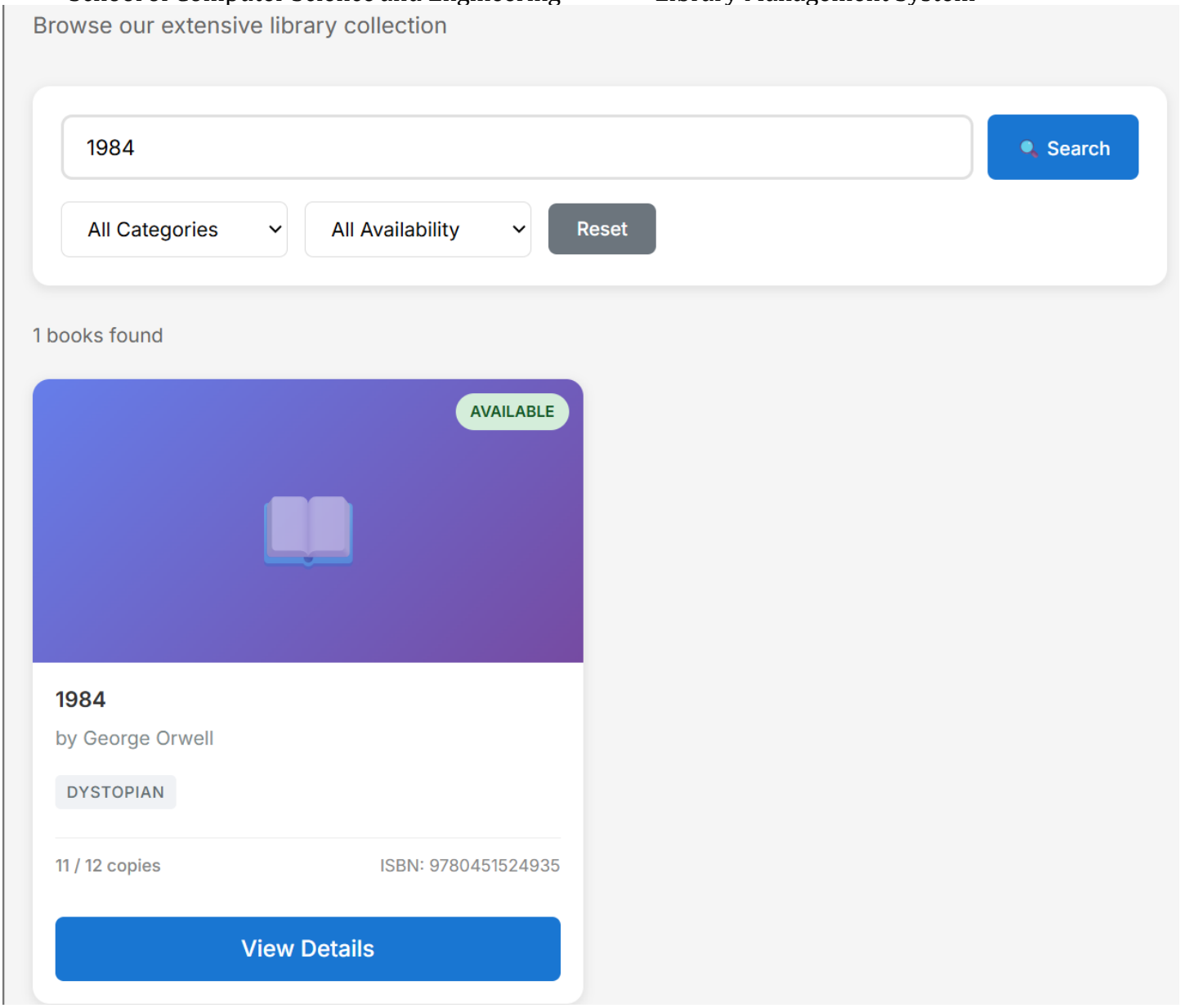

Figure 18 Result

- *The system will display a list of books matching your search criteria.*
- *Browse through the results to find the desired product.*

2.4 Borrow a Book

Step 1: *Log in to your account.*
Step 2: *Search for or browse to the book you wish to borrow.*
Step 3: *On the book's detail page, click the "Borrow" button (if the book is available).*
Step 4: *Confirm your borrowing request.*

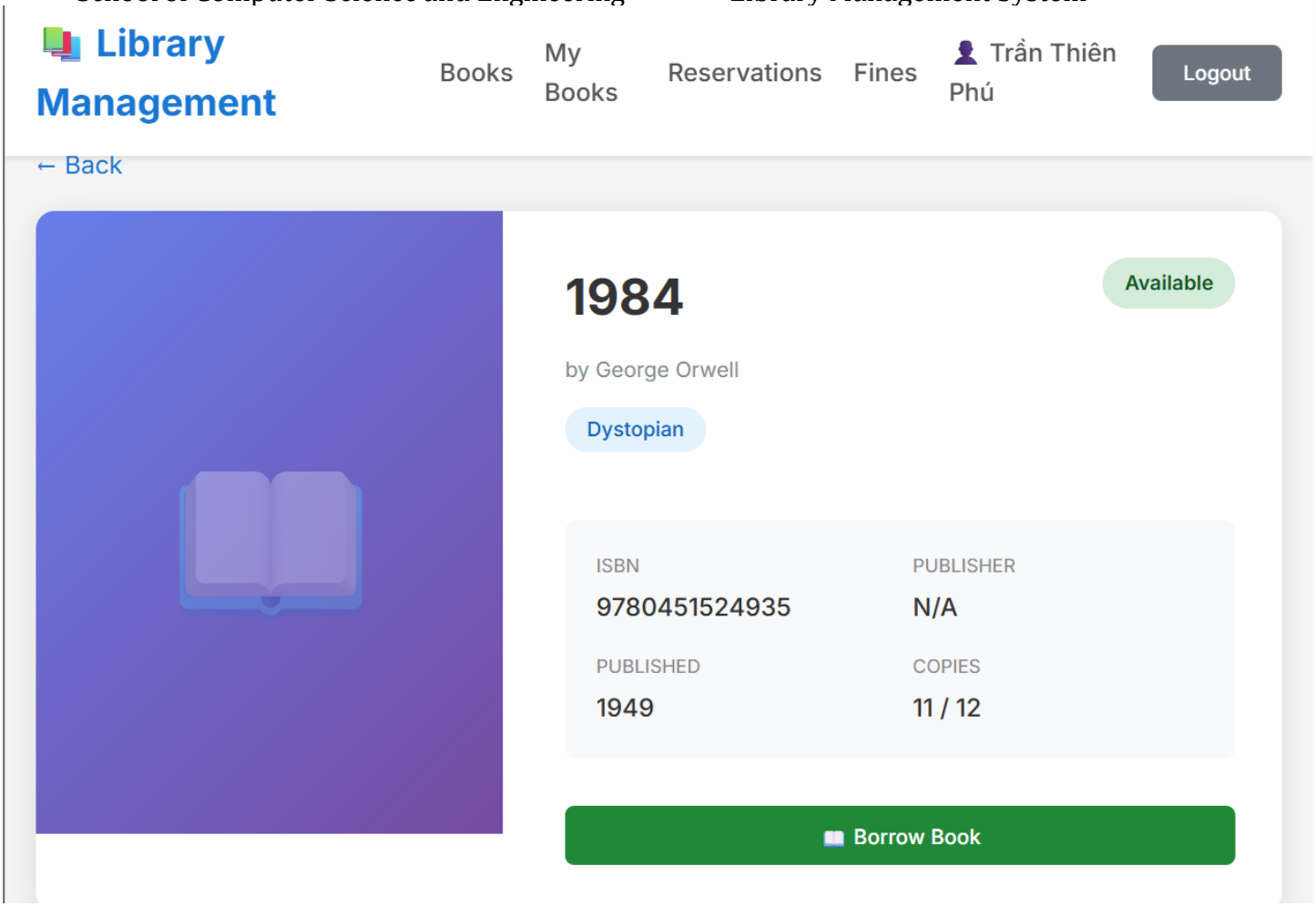

Figure 19 Borrow

Step 5: *The system will update the book's status and add the book to your borrowed list.*

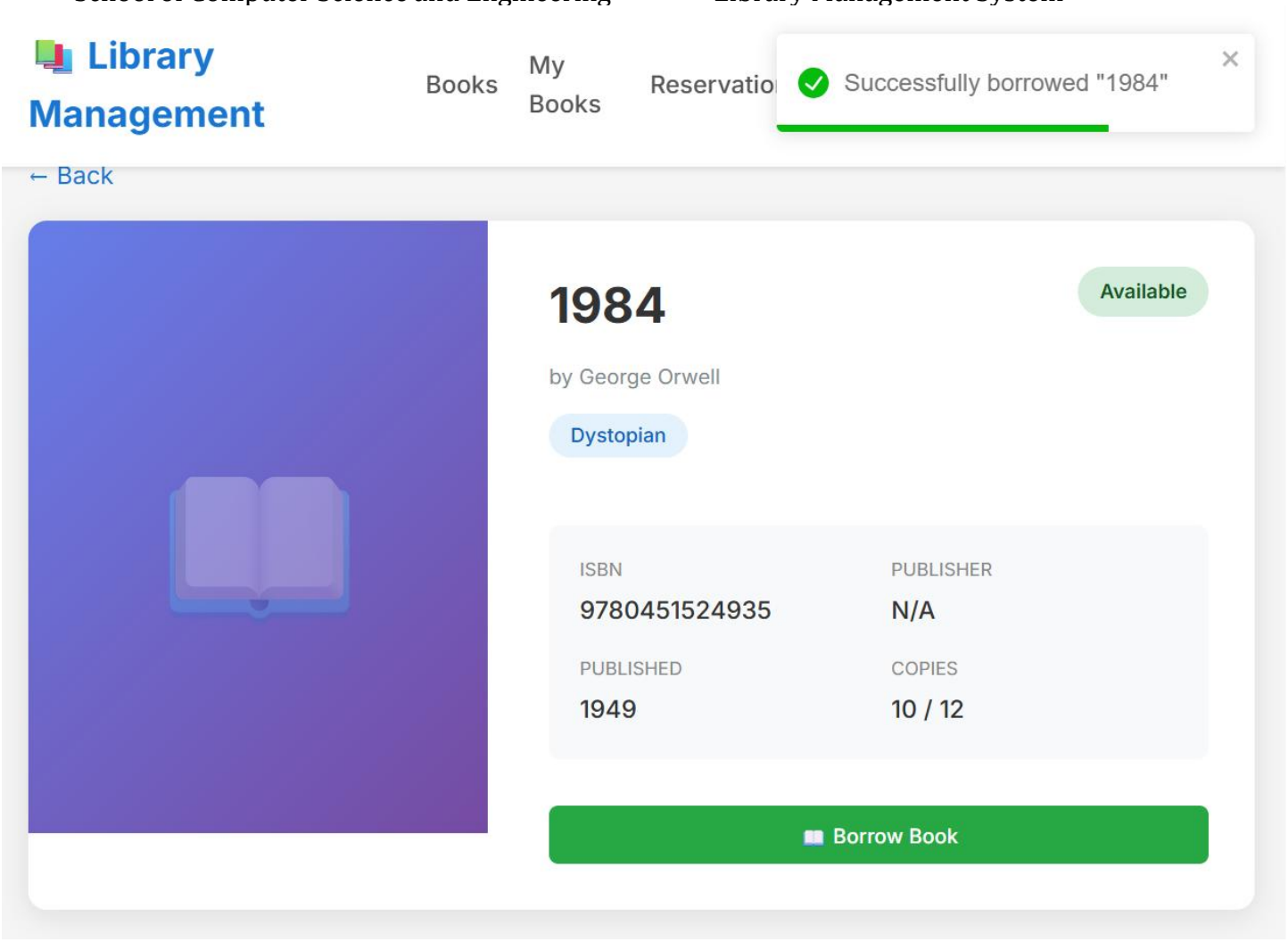

Figure 20 Result

Notes:

- *You may receive a due date for returning the book.*

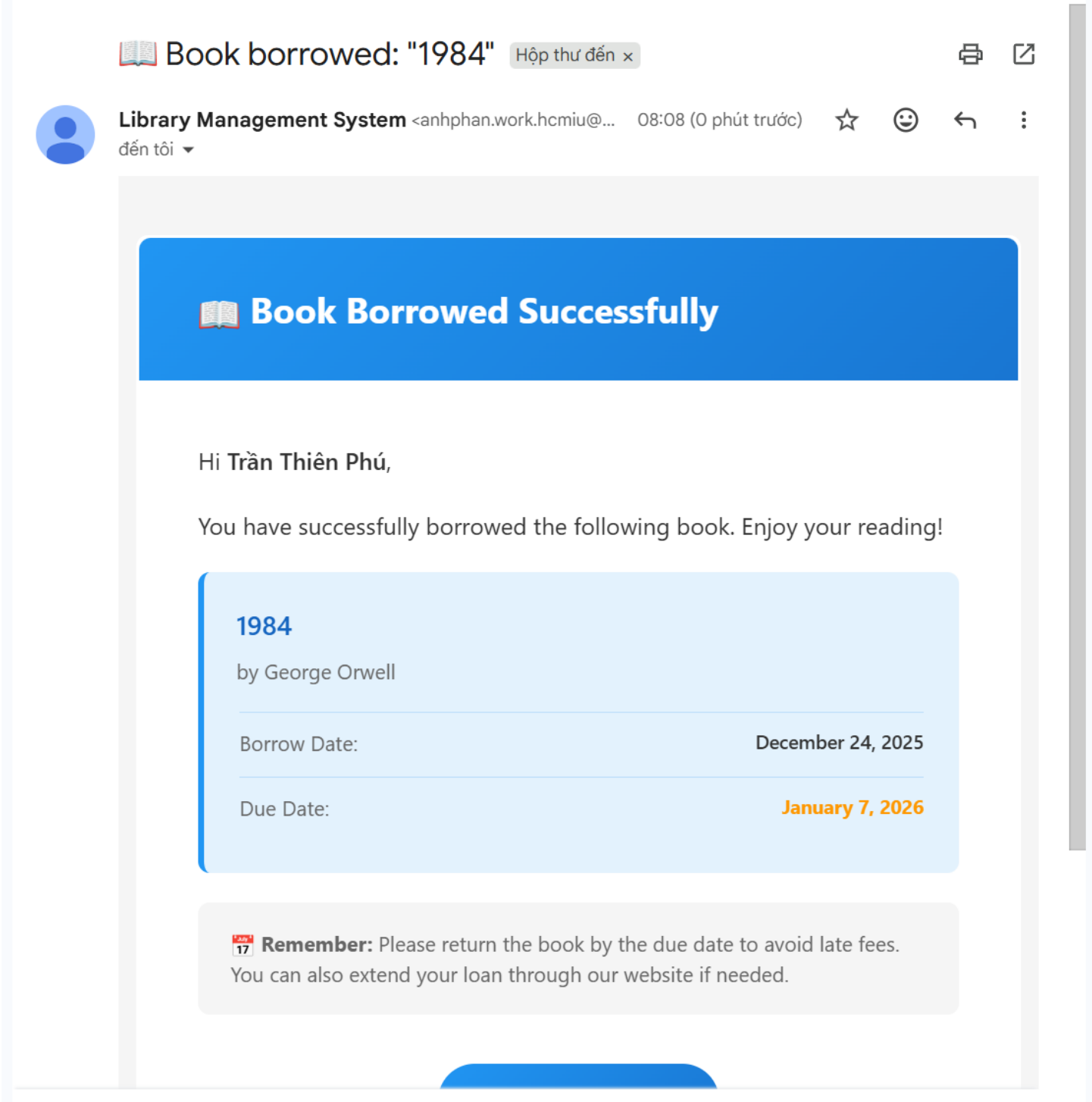

Figure 21 Borrow Email

- *If the book is not available, you may see a "Reserve" option instead.*

2.5 Reserve a Book

***Step 1**: Log in to your account.*

***Step 2**: Search for or browse to a book that is currently borrowed by someone else.*

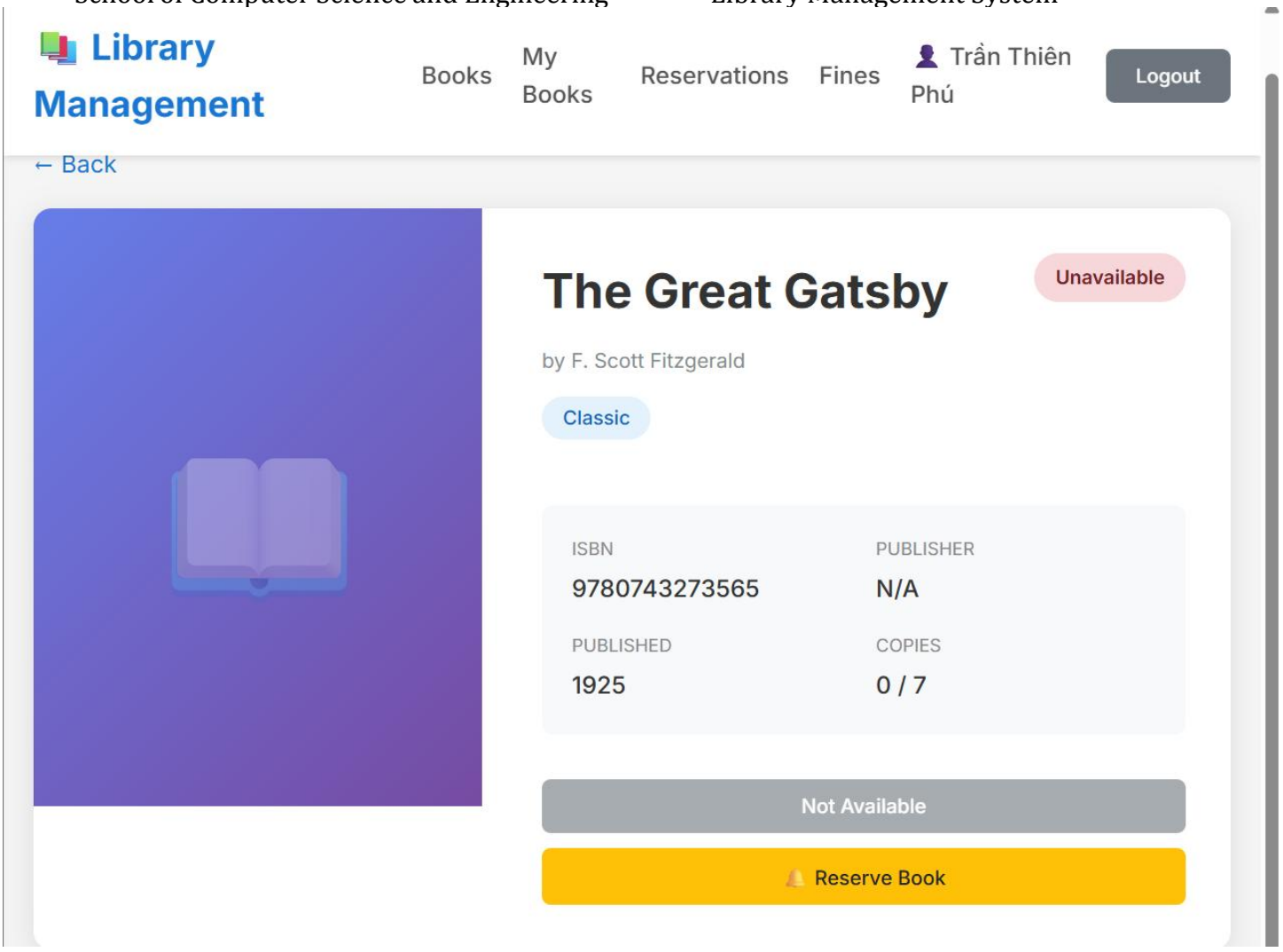

Figure 22 Reservation

***Step 3**: On the book's detail page, click the "Reserve" button.*

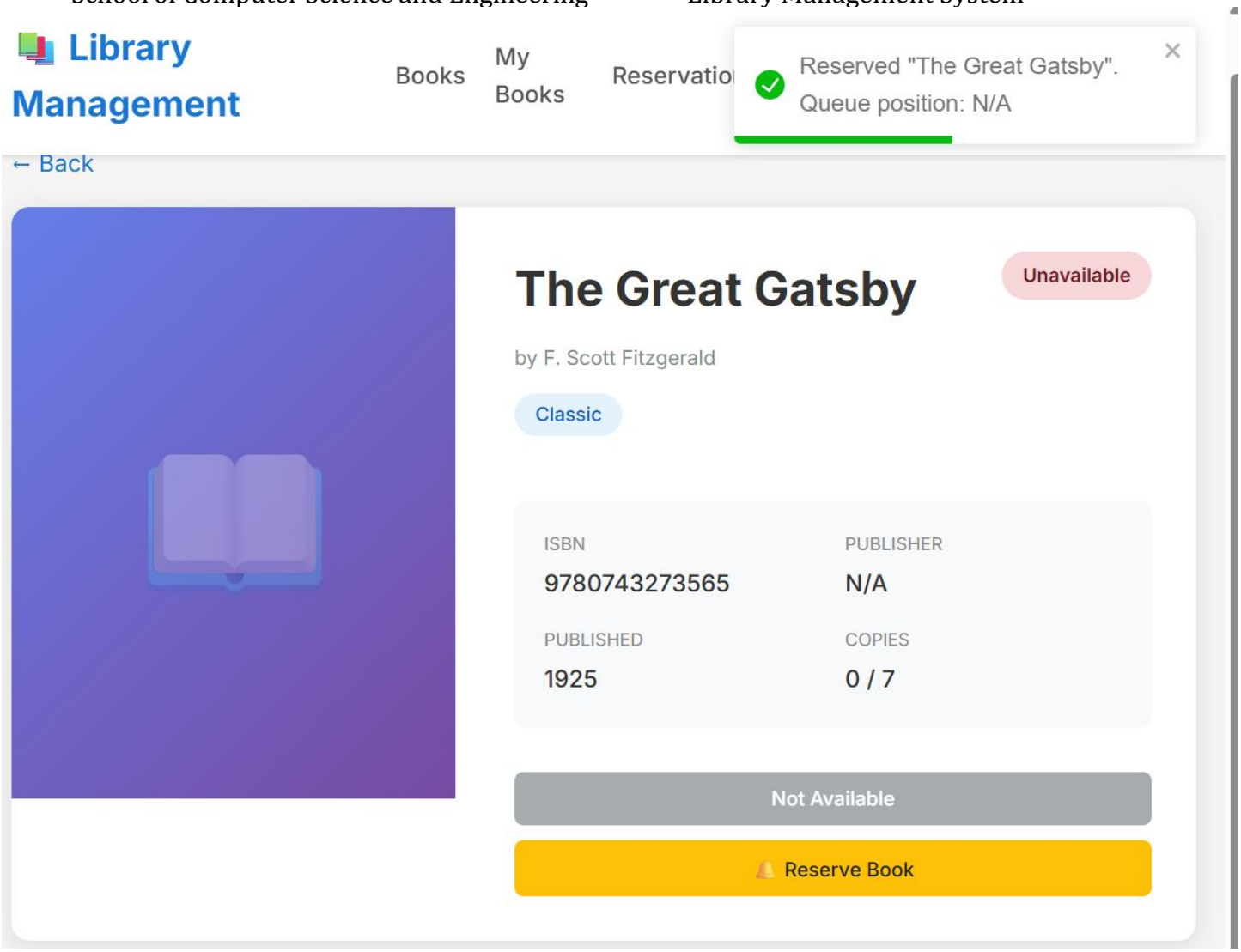

Figure 23 Reservation Result

***Step 4**: Confirm your reservation.*

***Step 5**: The system will notify you when the book becomes available.*

2.6 Return a Book

***Step 1:** Log in to your account.*
***Step 2:** Go to your **My books** list.*
***Step 3:** Select the book you wish to return and click the "Return" button.*

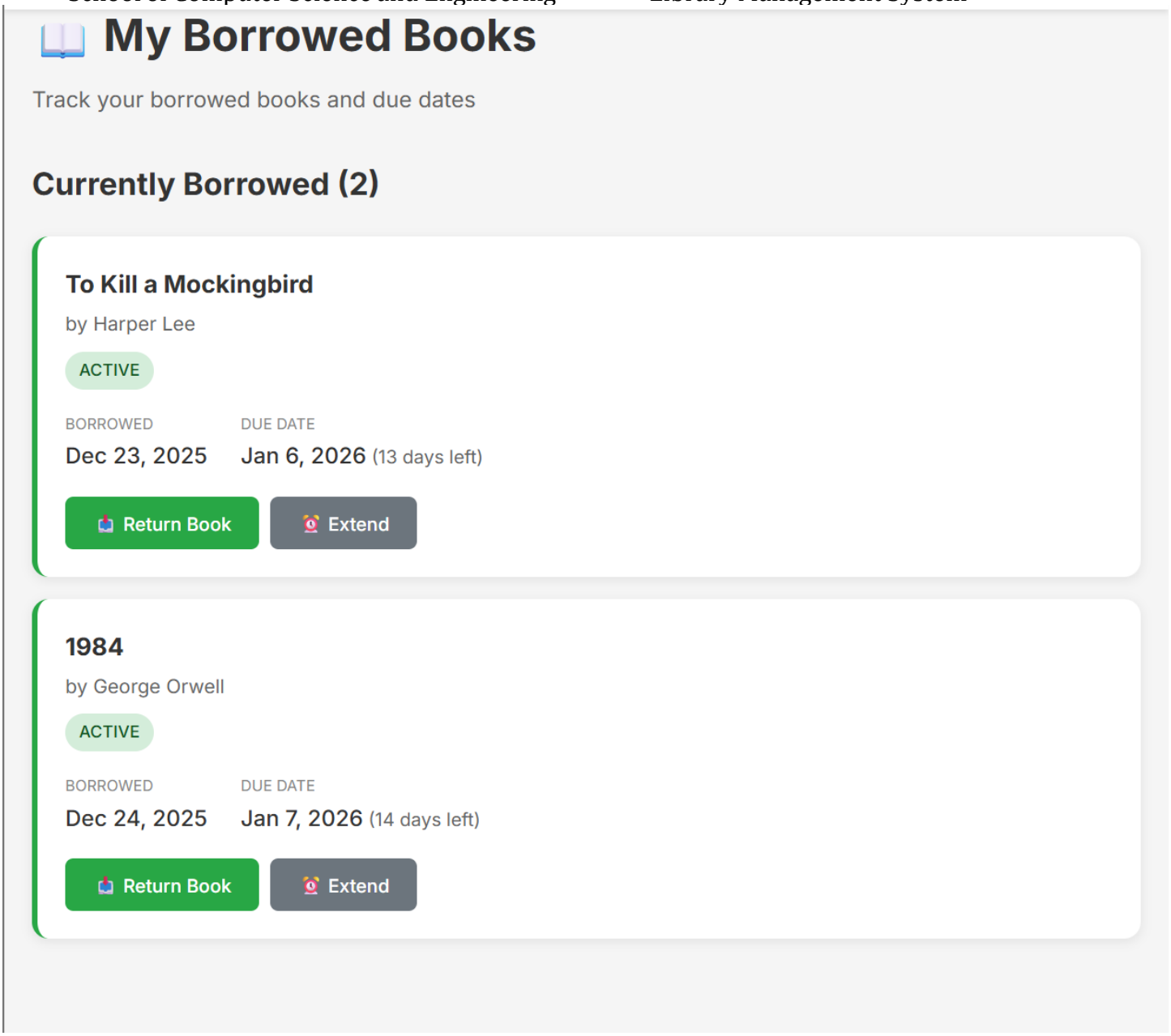

Figure 24 Return

Step 4: *Confirm the return.*
Step 5: *The system updates the book's status to available and removes it from your borrowed list.*

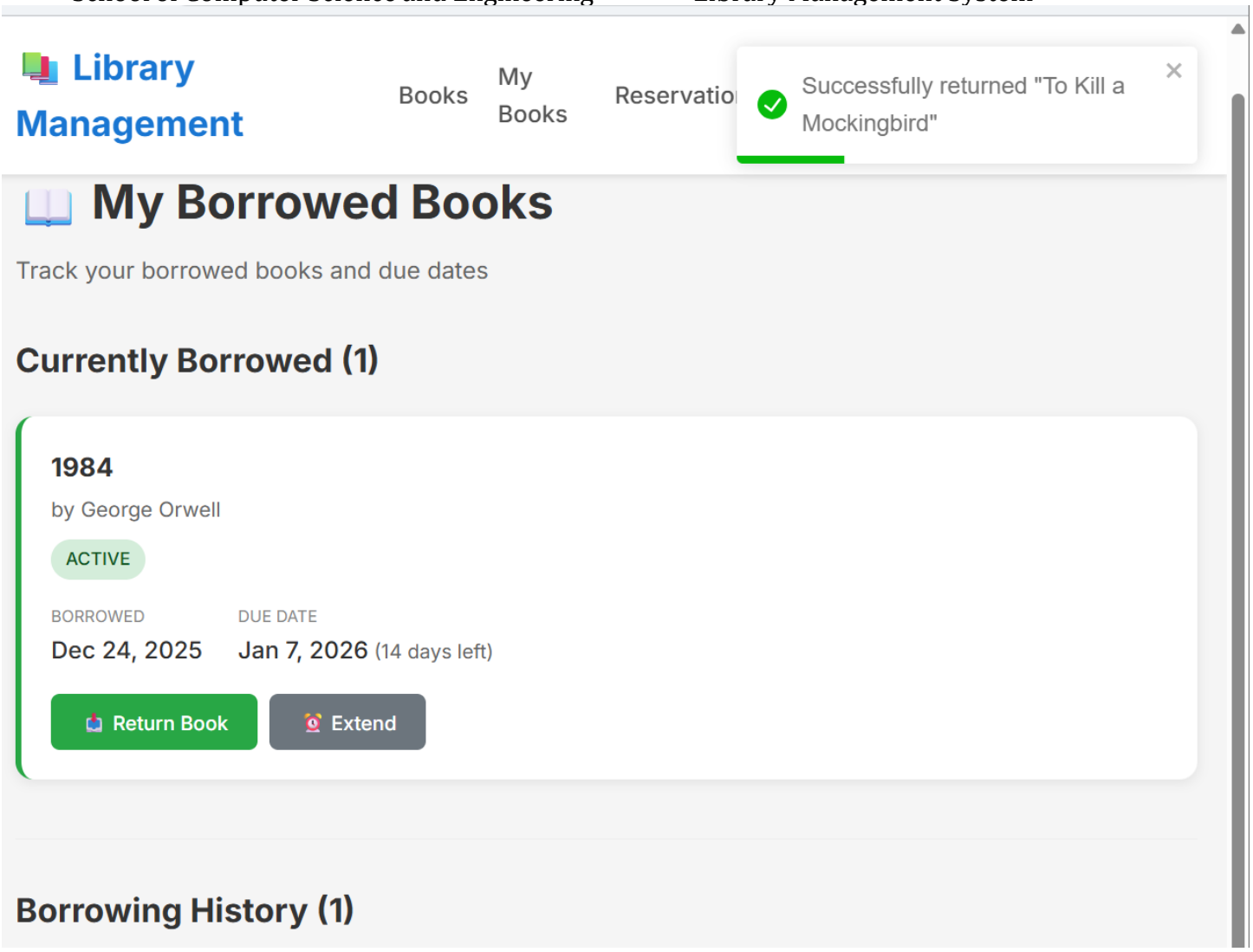

Figure 25 Return Result

System announces to your email.

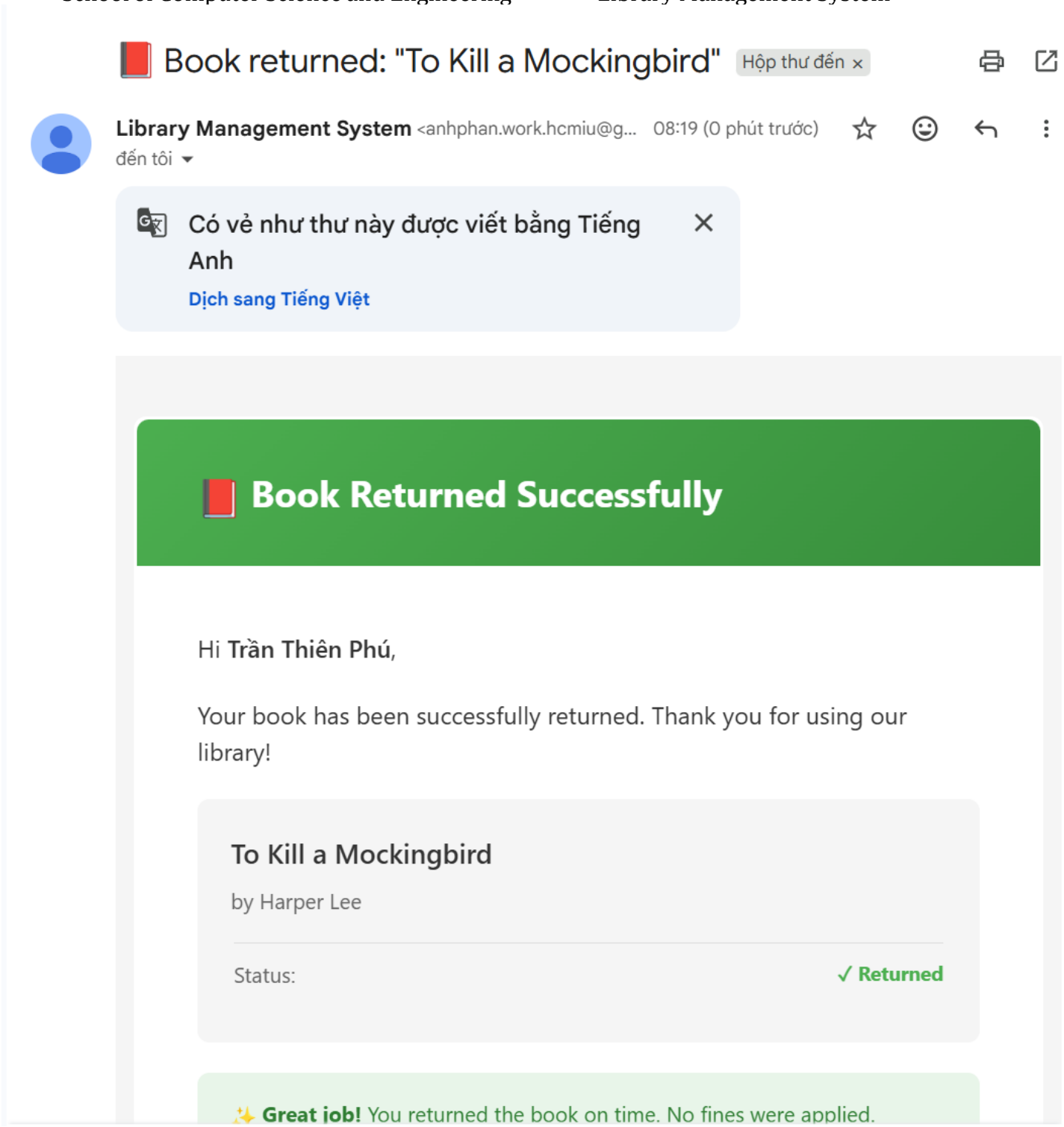

Figure 26 Return Email

Notes:

- *If the book is overdue, the system may calculate and display a fine.*

Admin's Function

1.1 Add a New Book

Step 1: *Log in to your admin account.*

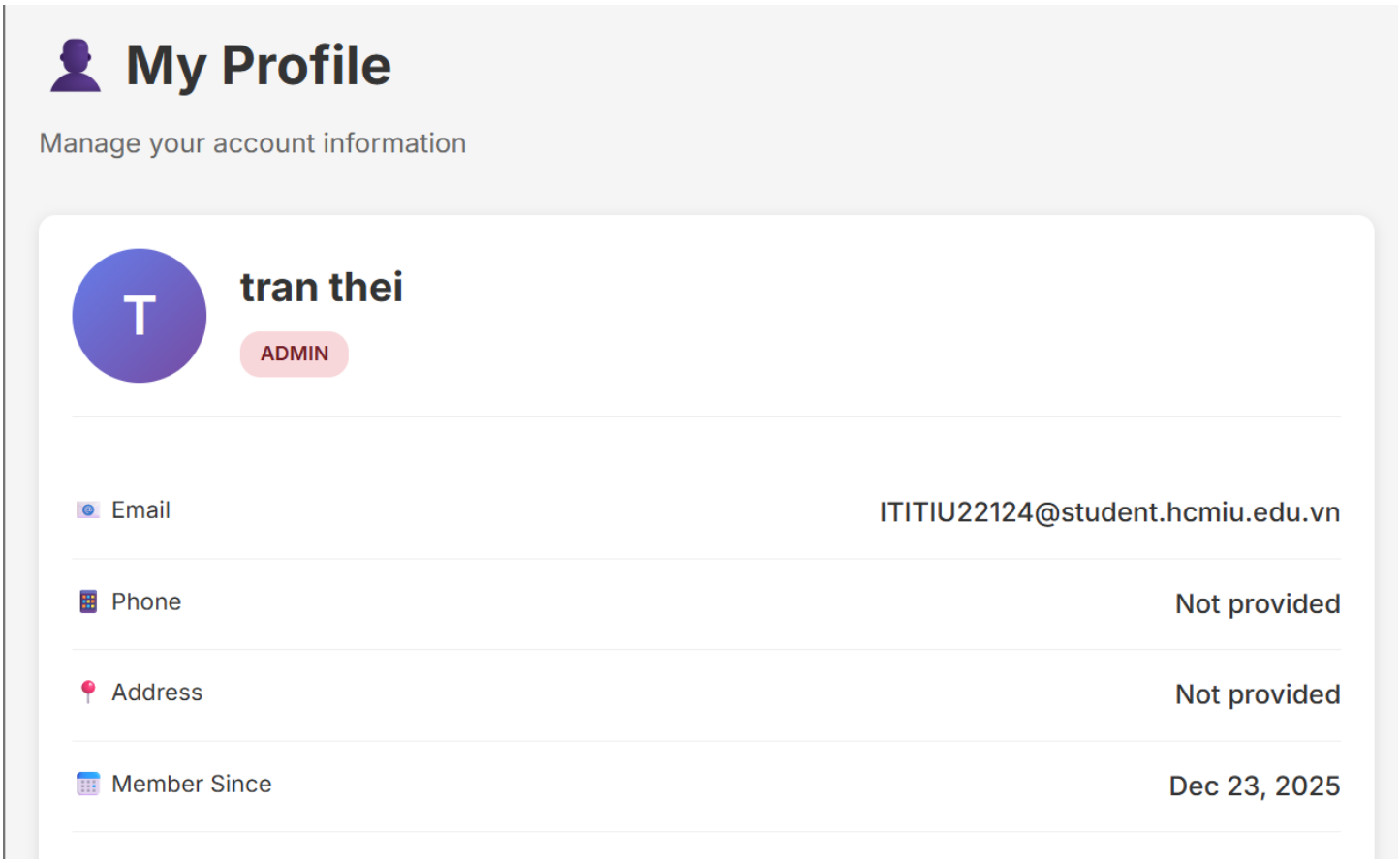

Figure 27 Admin Profile

Step 2: *Navigate to the "Admin DashBoard" section.*

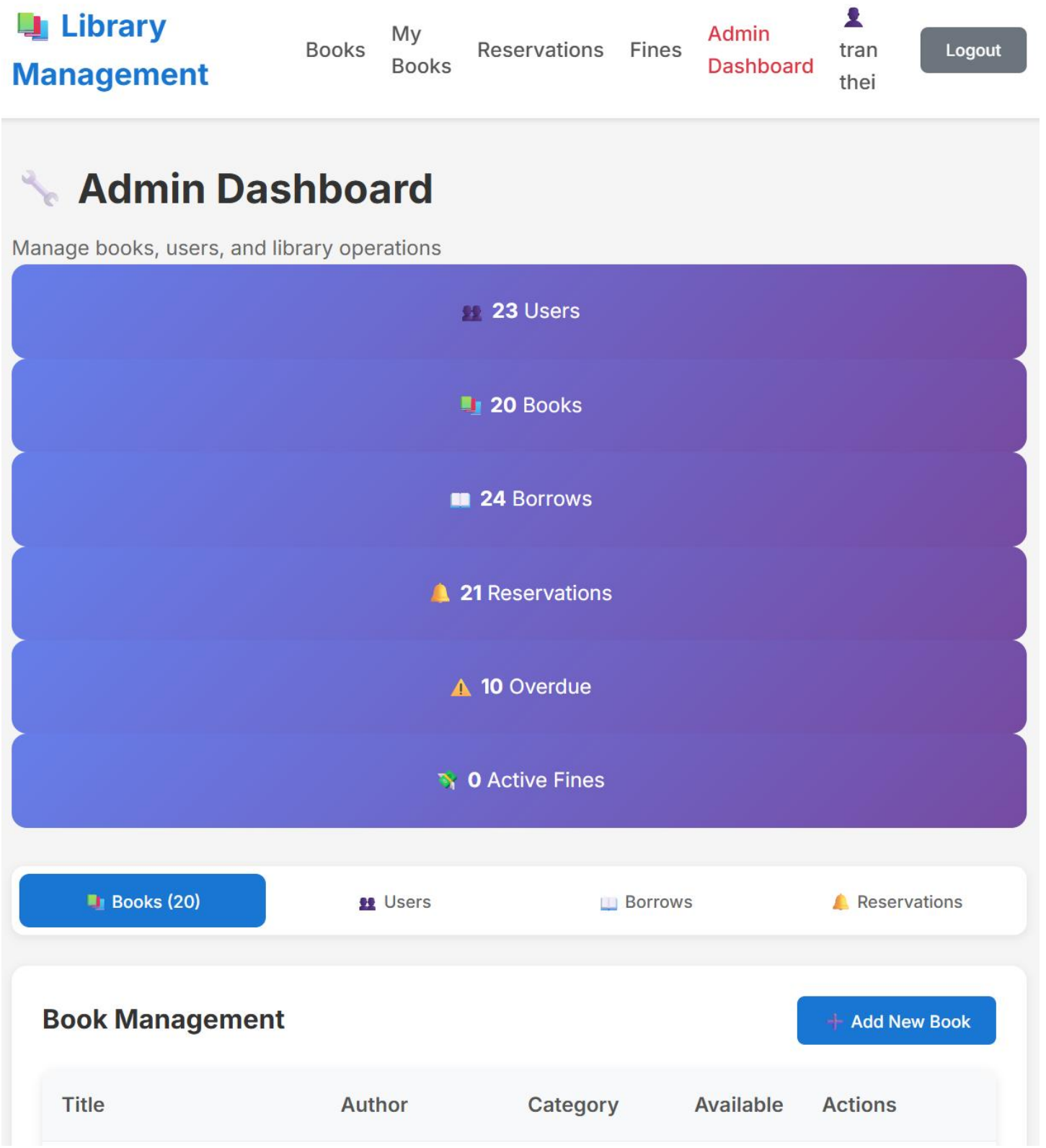

Figure 28 Admin DashBoard

Step 3: *Click the "Add New Book" button.*
Step 4: *Fill in the required book details (title, author, ISBN, category, status, etc.).*

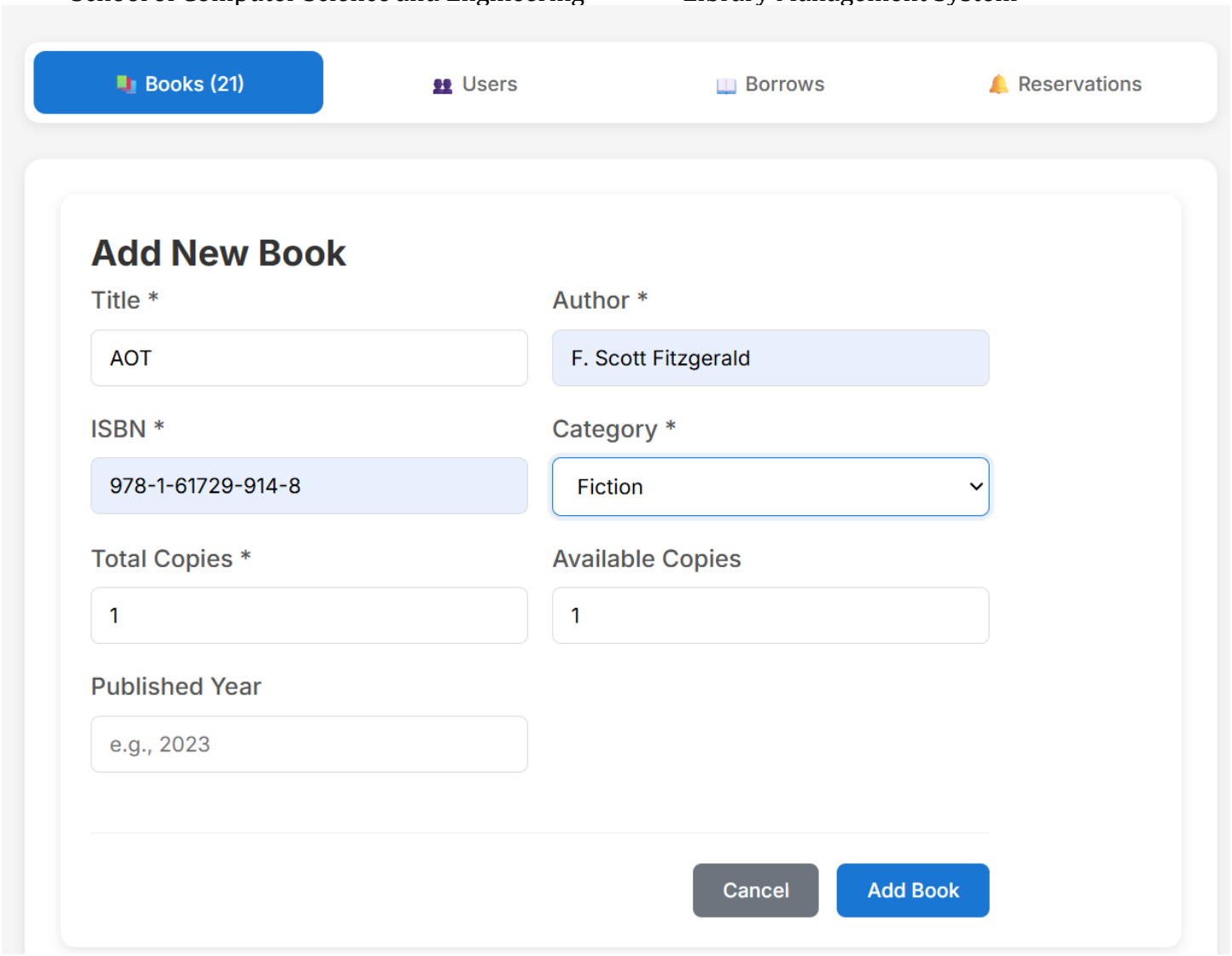

Figure 29 Add Book

Step 5: *Click "Add Book" to create the new book entry.*

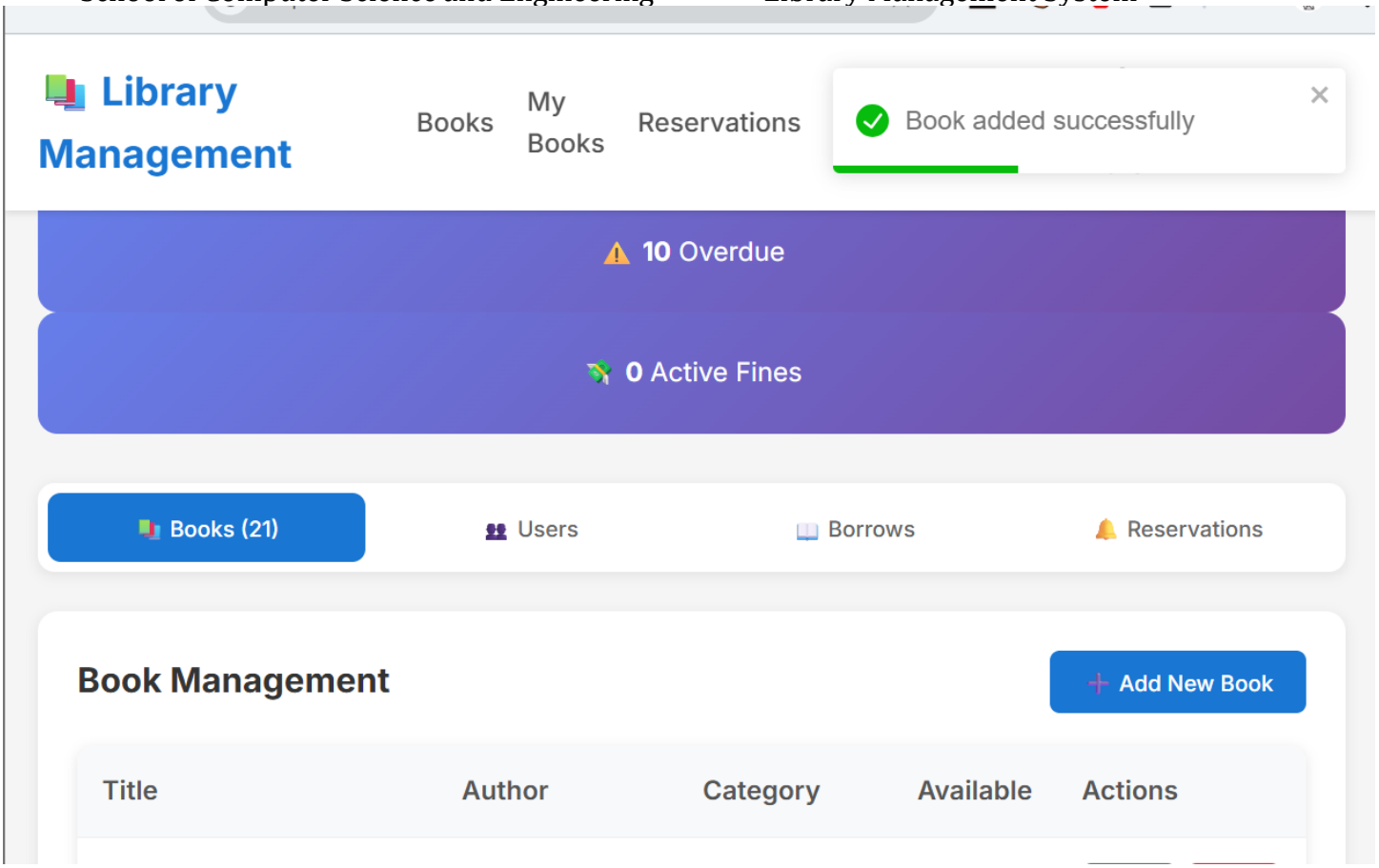

Figure 30 Add Book Result

Notes:

- *The new book will now appear in the catalog and be available for borrowing.*

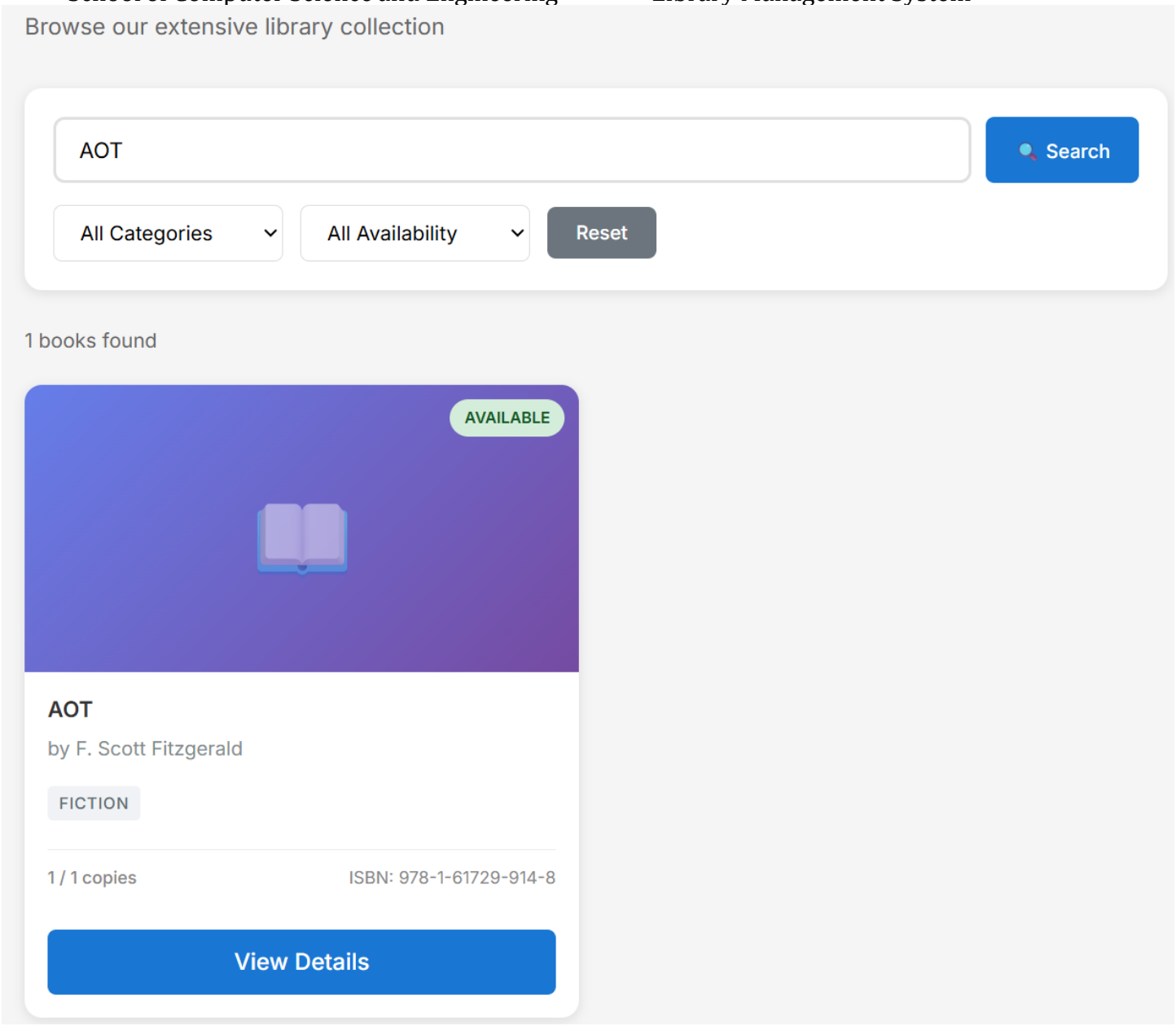

Figure 31 List Book

1.2 Update Or Edit Book Information

Step 1: *Log in to your admin account.*
Step 2: *Go to the "Books" section.*
Step 3: *Search for or select the book you wish to update.*

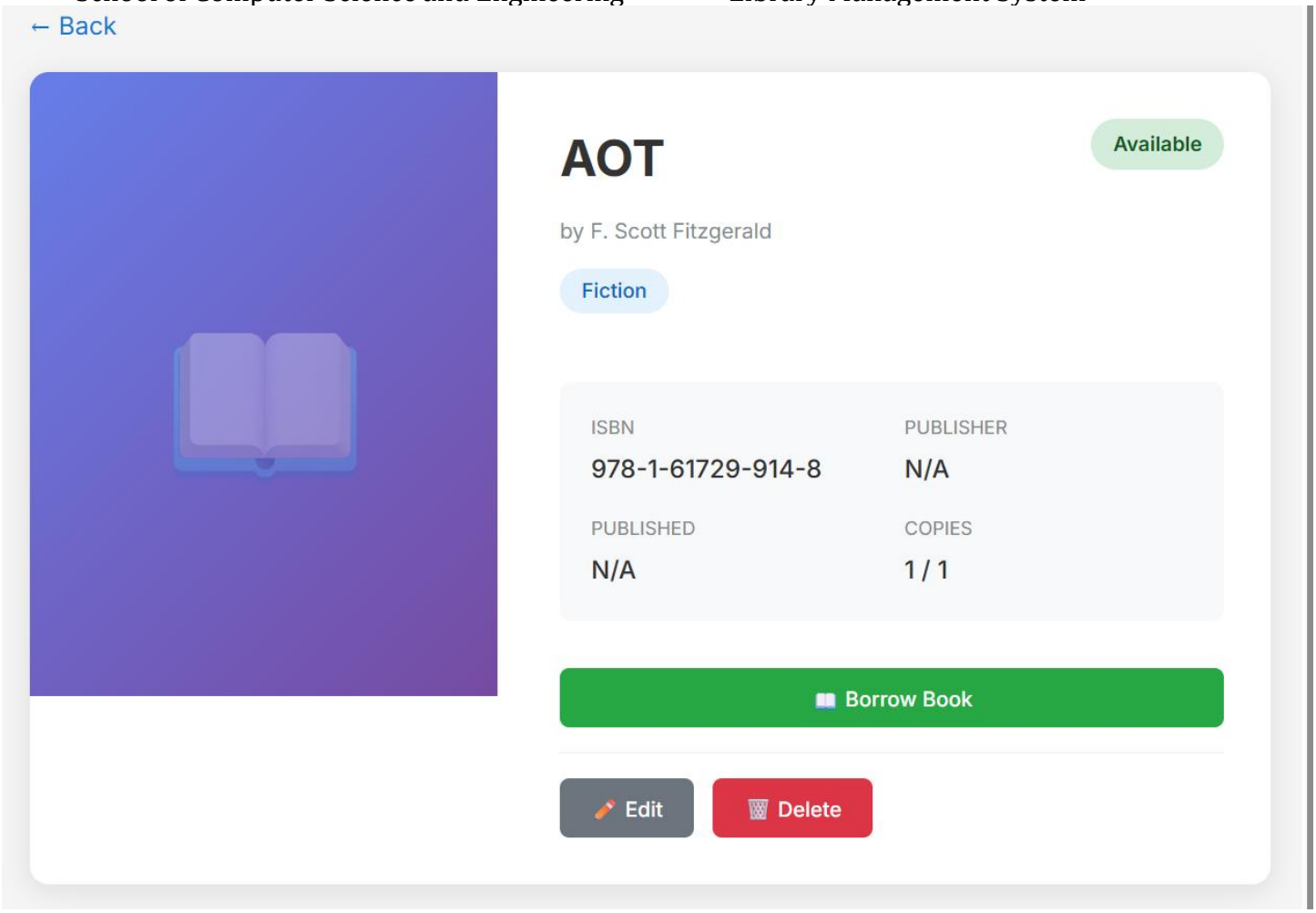

Figure 32 Edit or Update

Step 4: *Click the "Edit" or "Update" button next to the book.*
Step 5: *Modify the necessary details (e.g., title, author, status).*

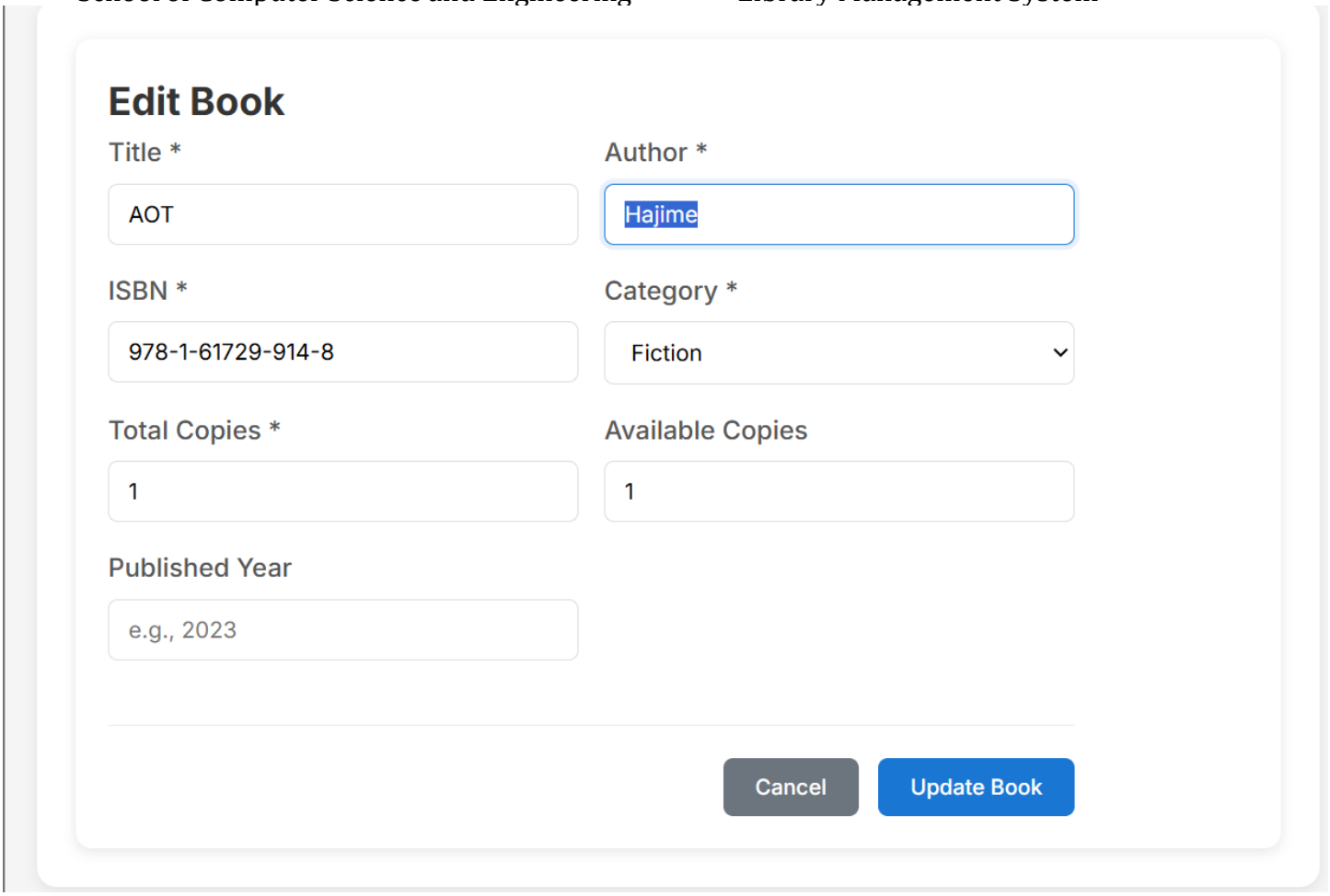

Figure 33 Edit or Update form

Step 6: *Click "Update Book" to apply the changes.*

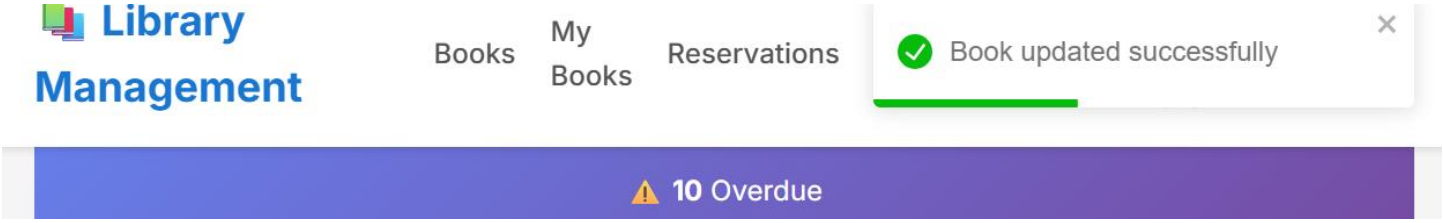

Figure 34 Result

Notes:

- *Updated information will be reflected immediately in the catalog.*

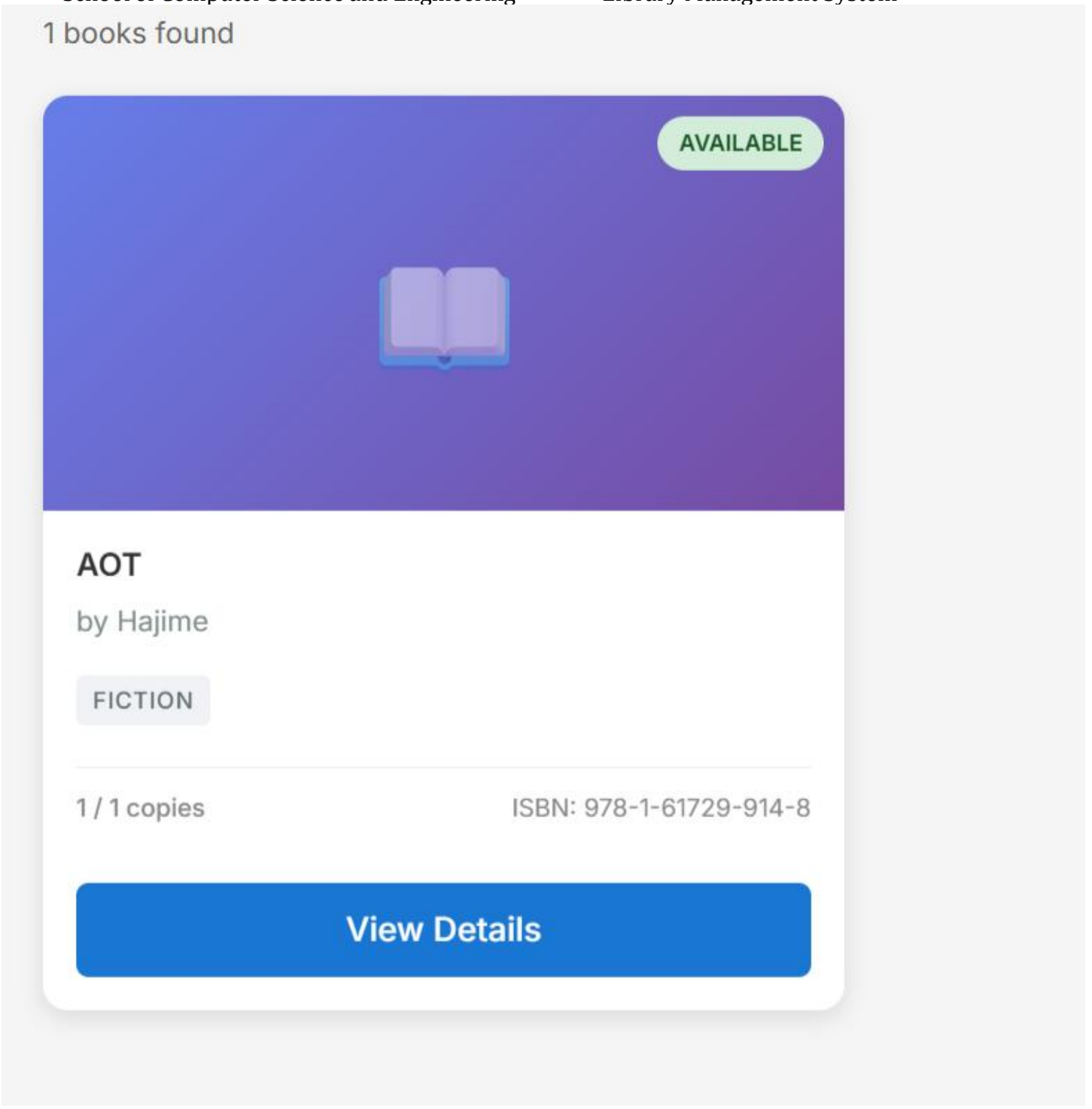

Figure 35 Updated Book

1.3 Delete a Book

Step 1: *Log in to your admin account.*
Step 2: *Navigate to the "Books" or "Catalog Management" section.*
Step 3: *Search for or select the book you wish to delete.*

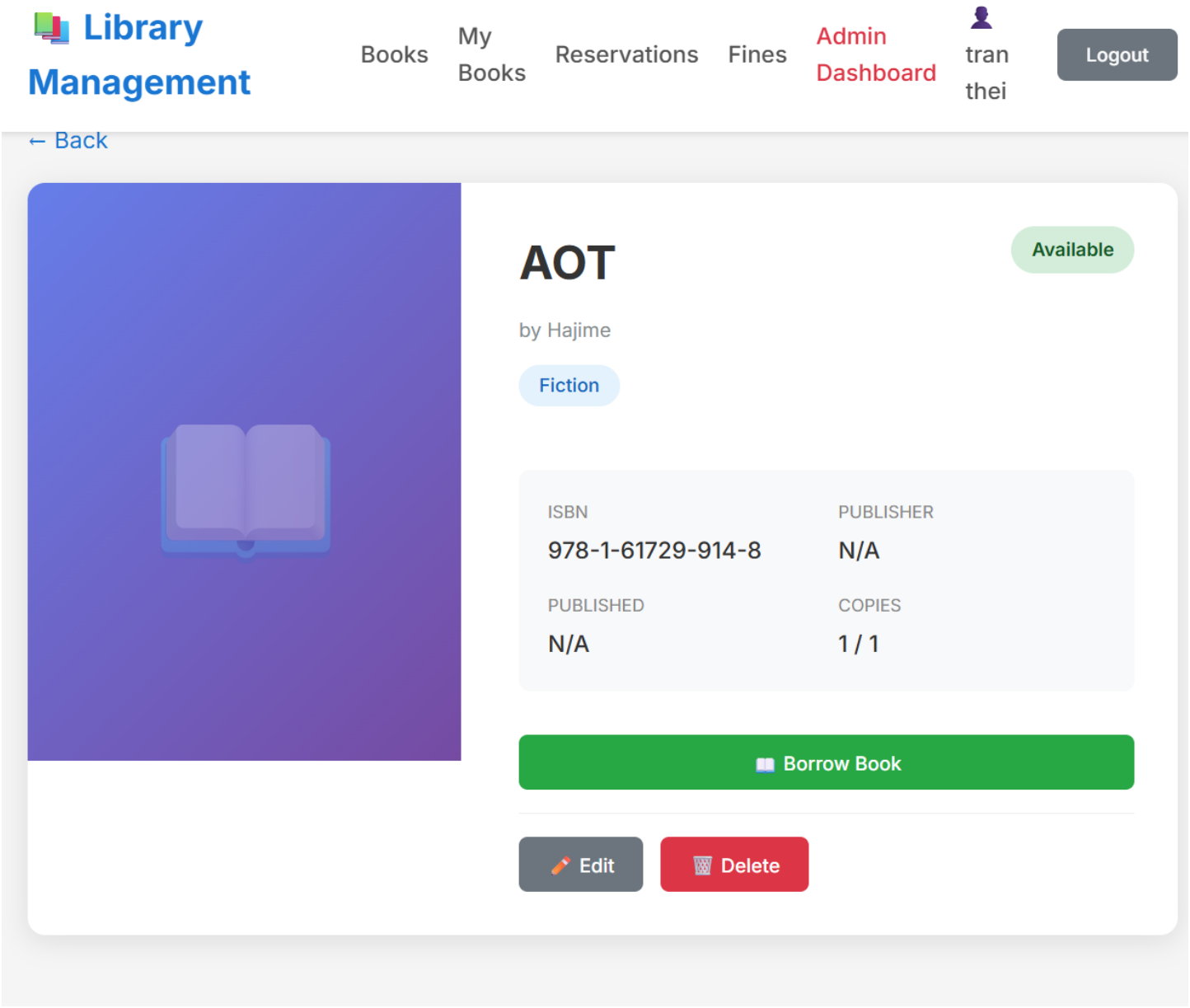

Figure 36 Delete

Step 4: *Click the "Delete" button next to the book.*
Step 5: *Confirm the deletion when prompted.*

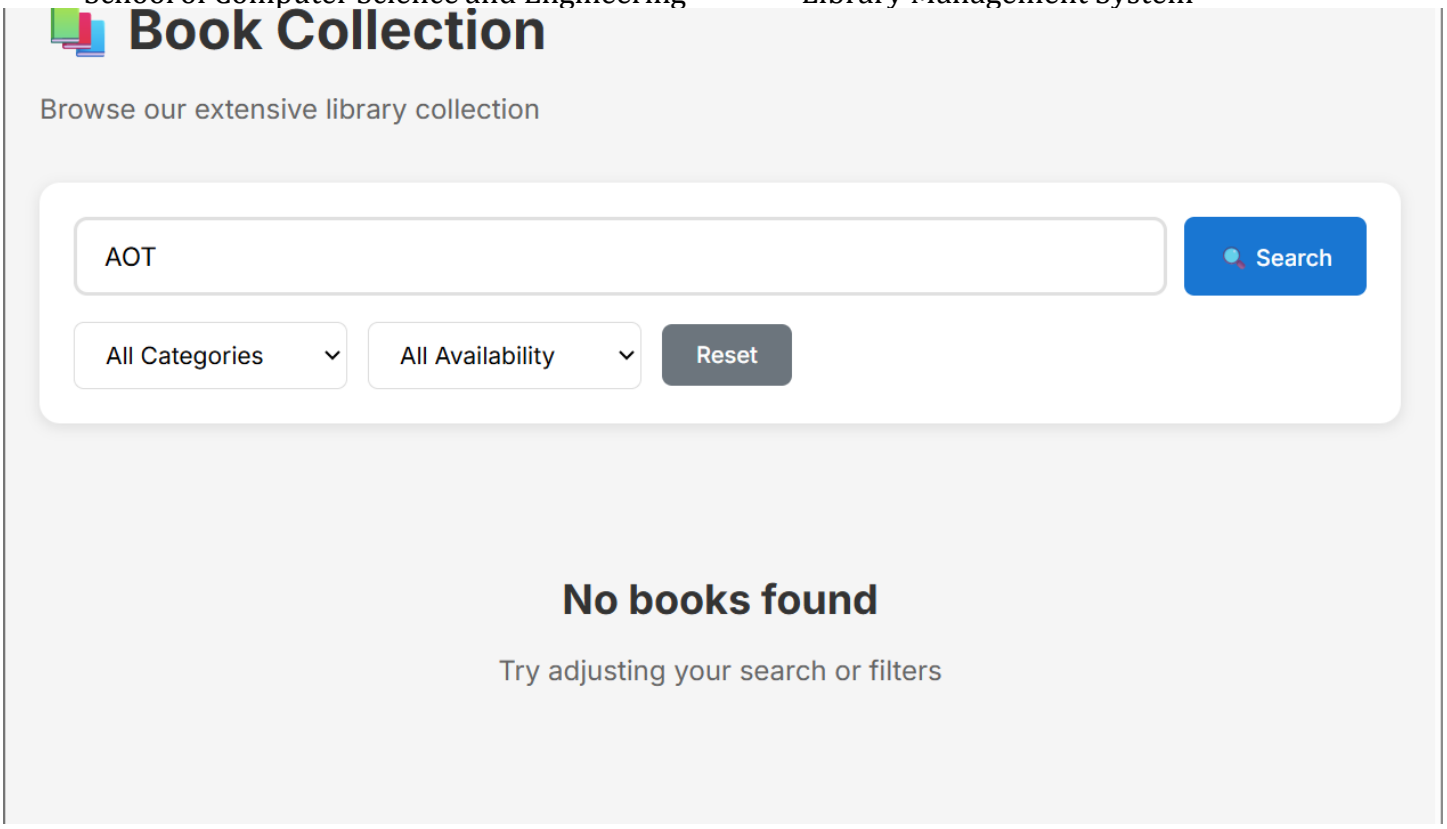

Figure 37 Result

Notes:

- *Deleted books will be removed from the catalog and cannot be borrowed.*

Admin Management

2.1Manage Users

Step 1: Log in to your admin account.
Step 2: Go to the "Admin Dashboard" section.

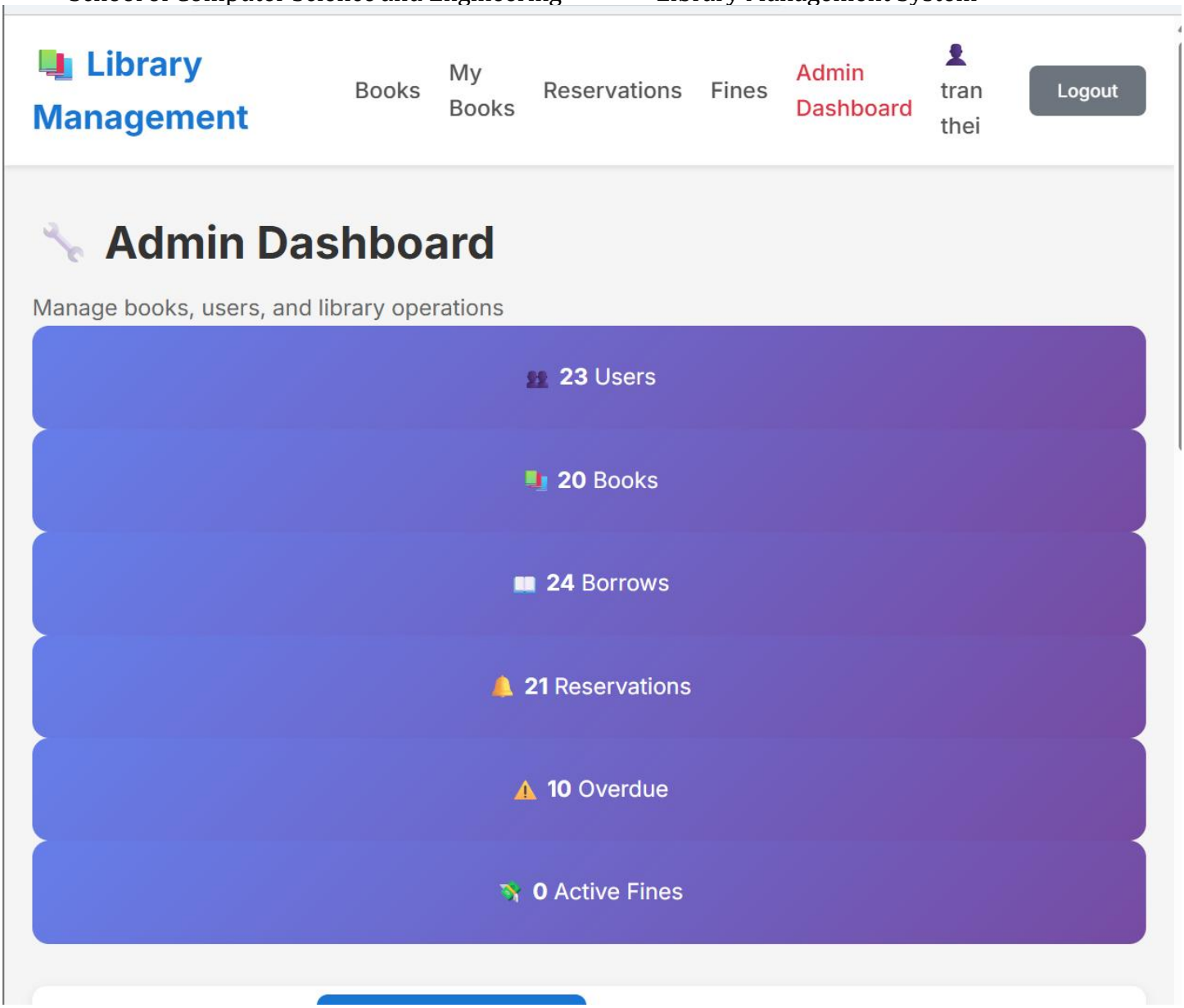

Figure 38 Admin Dashboard

Step 3: View the list of registered users.

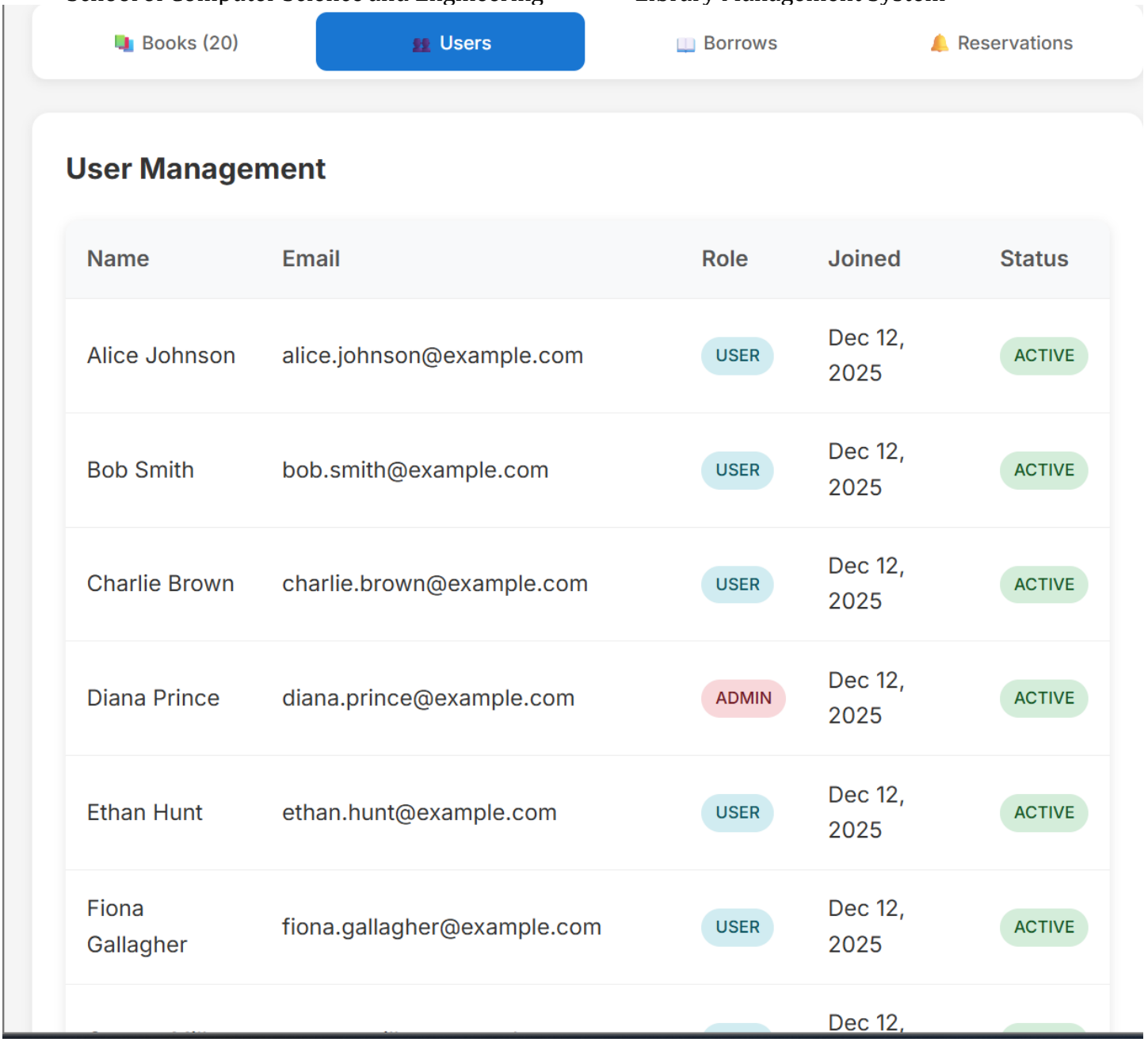

Figure 39 User Report

2.2 View Reports

Step 1: Log in to your admin profile.
Step 2: "View Report."

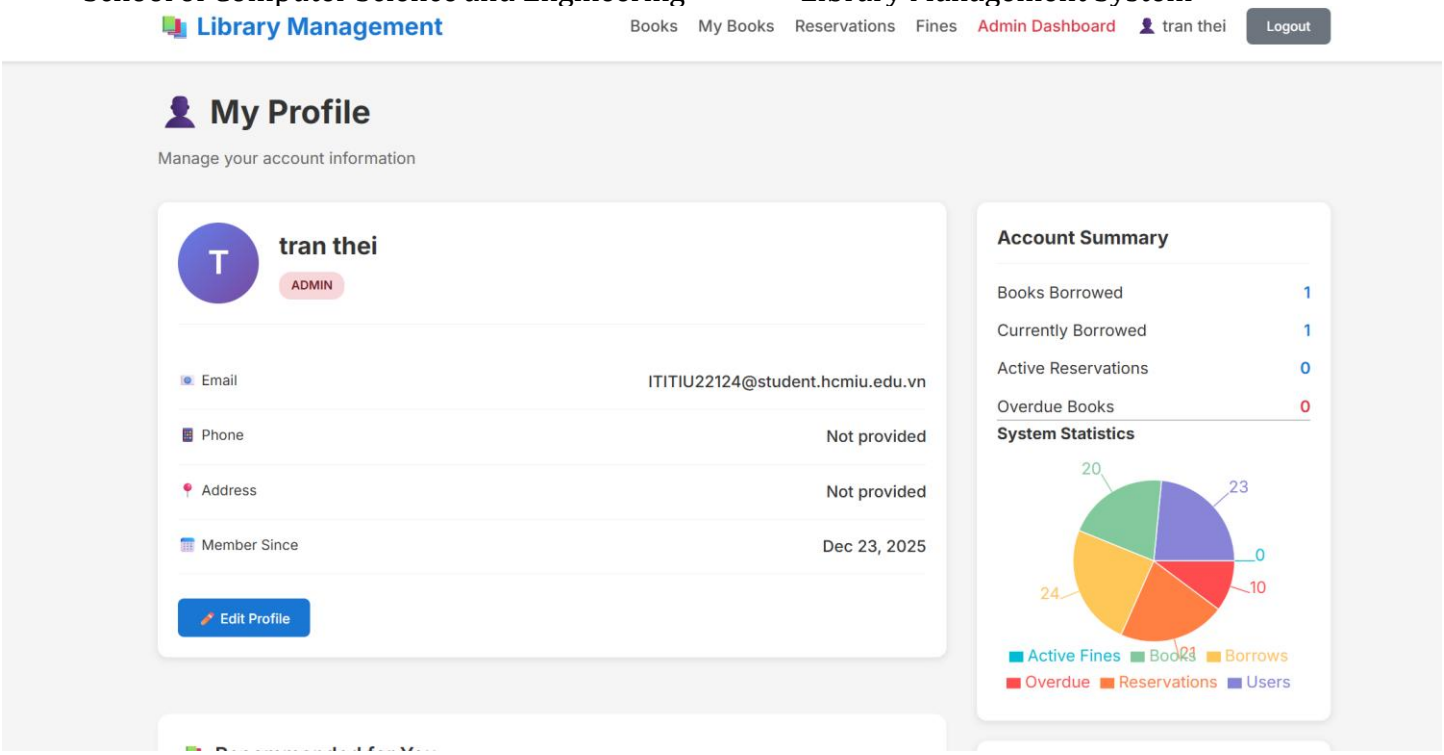

Figure 40 Admin Profile

2.3 Monitor Library Activity

Step 1: *Log in to your admin account.*
Step 2: *Go to the "Admin Dashboard" section.*
Step 3: *Review recent activities such as borrow/return events, reservations, and user registrations.*

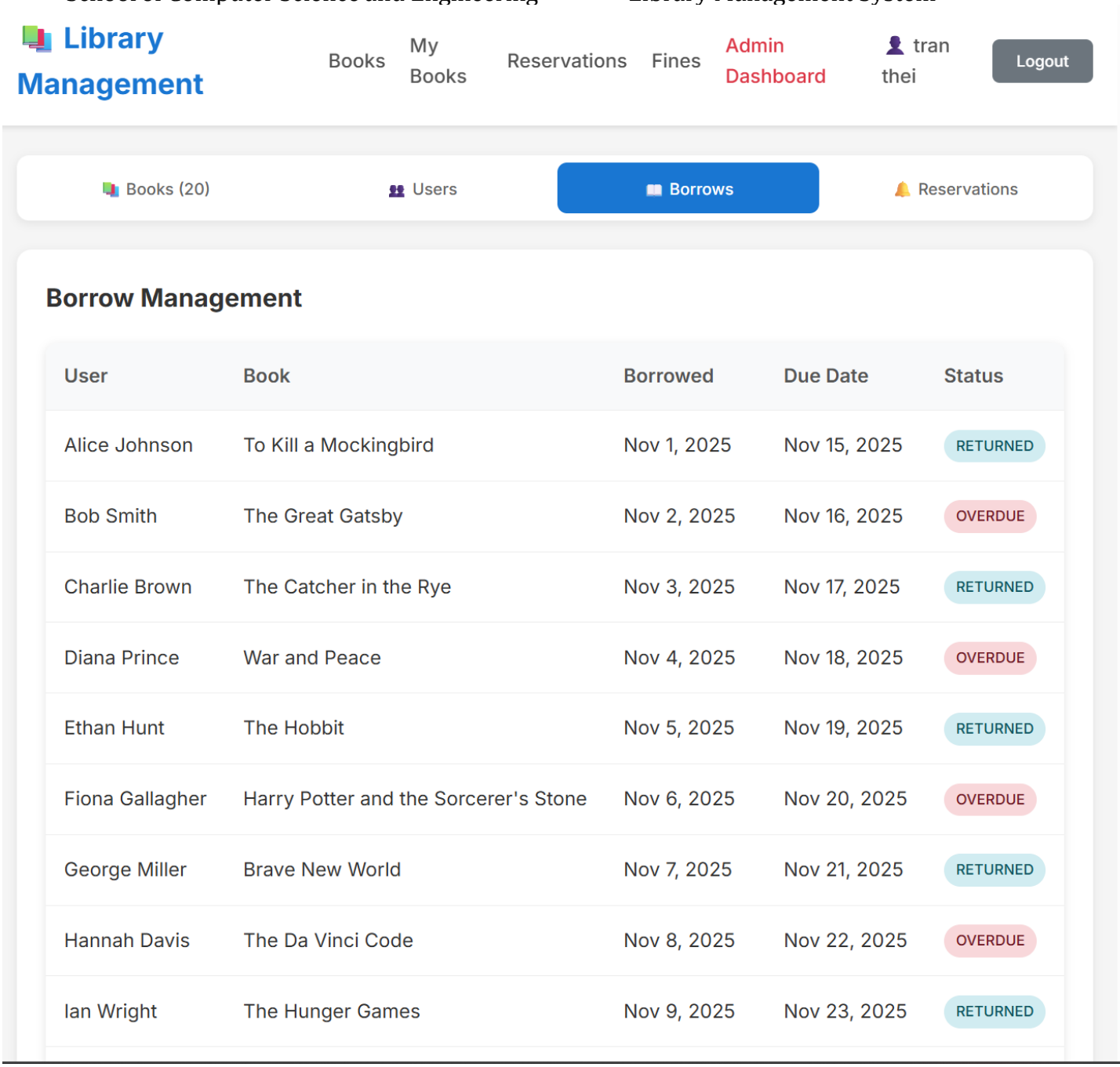

Figure 41 Monitor Borrows

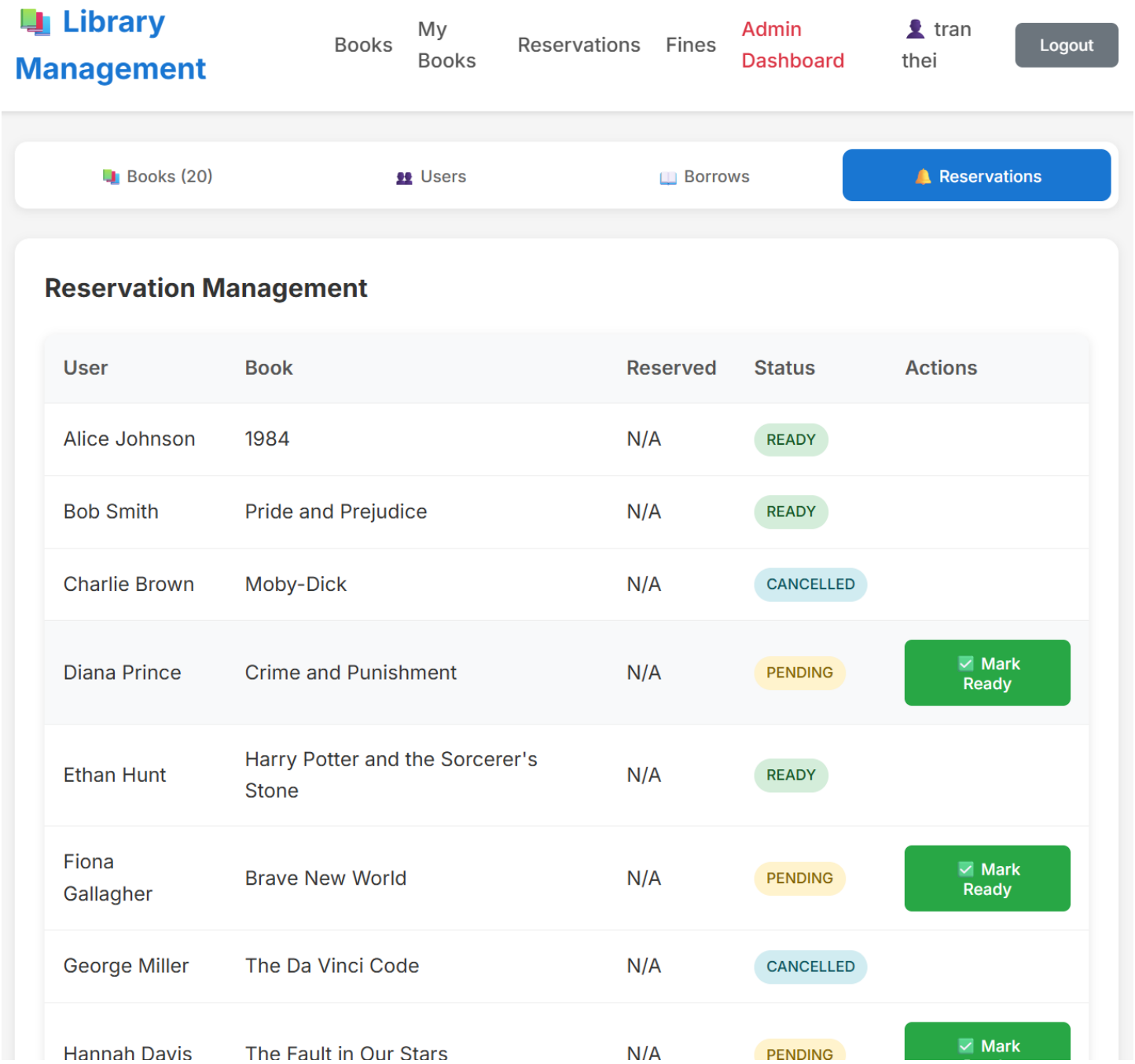

Figure 42 Motitor Reservations

- ***Monitoring helps identify trends and potential issues in library operations.***

III.CONCLUSION

The Library Management System project has successfully met its objectives by delivering a comprehensive platform for managing library operations. The system enables seamless book cataloging, user management, borrowing, reservations, and fine processing, all through an intuitive web interface. The integration of a React.js frontend with a Spring Boot

backend ensures a responsive user experience and robust server-side logic. Automated email notifications, secure authentication, and role-based access control further enhance the system's reliability and usability.

Achievements

- *Developed a modular, maintainable codebase using industry-standard frameworks.*
- *Implemented core library features: book management, borrowing/returning, reservations, and fines.*
- *Provided a modern, user-friendly interface for both users and administrators.*
- *Ensured data integrity and security through validation and access control.*
- *Automated user communications with dynamic email notifications.*
- *Facilitated easy deployment and testing with scripts and documentation.*

Future Work

- *Expand reporting and analytics for library staff (e.g., usage trends, overdue statistics).*
- *Integrate barcode or RFID support for faster book check-in/out.*
- *Add support for mobile devices and develop a dedicated mobile app.*
- *Implement recommendation systems for users based on borrowing history.*
- *Enable integration with external library systems or digital content providers.*
- *Enhance accessibility features for users with disabilities.*
- *Support for multiple branches or locations within the same system.*

Reflections

This project provided valuable experience in full-stack development, system integration, and agile project management. Challenges such as synchronizing frontend-backend communication, ensuring data consistency, and designing for scalability were overcome through iterative development and testing. The collaborative process fostered problem-solving and adaptability, while user feedback guided improvements in usability and functionality. Looking ahead, the system's flexible architecture will support future enhancements and adaptation to evolving library needs

IV.REFERENCES

1. Liu, Y. (n.d.). Web Application Security: Threats, Countermeasures, and Pitfalls. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050920303772
2. Spring Boot Reference Documentation. (2025). Retrieved from https://docs.spring.io/spring-boot/docs/current/reference/html/
3. React Documentation – A JavaScript library for building user interfaces. (2025). Retrieved from https://react.dev/
4. Mozilla Developer Network (MDN). (2025). Web APIs and Frontend Development. Retrieved from https://developer.mozilla.org/
5. MySQL 8.0 Reference Manual. (2025). Retrieved from https://dev.mysql.com/doc/refman/8.0/en/
6. OWASP Foundation. (2025). OWASP Top Ten Web Application Security Risks. Retrieved from https://owasp.org/www-project-top-ten/
7. Bootstrap Documentation. (2025). Retrieved from https://getbootstrap.com/docs/
8. GitHub Copilot. (2025). AI-powered code completion and documentation. Retrieved from https://github.com/features/copilot