Library Management System

The system to be implemented is a comprehensive Library Management System (LMS) designed to facilitate the management of library resources and operations. The LMS will allow users to search for books, borrow them, and manage their borrowing history. It will also enable library administrators to manage book inventories, user accounts, and borrowing transactions. The system will be built using Java for the backend, with a SQL Server database to store data related to books, users, borrowing details, and categories.

Key Features:

1. User Management: Users can register, log in, and update their profiles. The system will differentiate between regular users and administrators based on roles.

Book Management: Administrators can add, update, and delete books. Books are categorized, and each book has attributes like ISBN, author, publisher, and quantity.

Borrowing System: Users can borrow books, and the system will track borrowing details, including the borrow and return dates. Users can view their borrowing history and cancel pending borrow requests.

Search and Filter: Users can search for books by title, author, or ISBN and filter them by category.

Role-Based Access Control: Different functionalities are accessible based on user roles, ensuring that only authorized users can perform certain actions, such as managing books or users.

Business Rules:

Users must be registered and logged in to borrow books.

A book can only be borrowed if it is available in the library's inventory.

Borrowing requests can be canceled only if they are in a pending state.

Administrators have the exclusive right to manage book inventories and user accounts.

The database design for the FuLibrary system is structured to efficiently manage library operations, including book management, user management, and borrowing transactions. Below is an interpretation of the logical and physical design based on the provided SQL script.

Logical Design

The logical design outlines the entities, their attributes, and the relationships between them. The key entities in the FuLibrary database are:

1. Book

Attributes: ID, bookname, ISBN, edition, author, publisher, yearofrelease, physicaldescription, categoryId, Image, status, quantity

Primary Key: ID

Foreign Key: categoryId references BookCategory(ID)

2. BookBorrow

Attributes: ID, bookId, userId, borrowdetailId, createdAt, status

Primary Key: ID

Foreign Keys:

bookId references Book(ID)

userId references Users(ID)

borrowdetailId references BorrowDetail(ID)

BookCategory

Attributes: ID, categoryName

Primary Key: ID

4. BorrowDetail

Attributes: ID, borrowFrom, borrowTo, createdAt, nota

Primary Key: ID

Role

Attributes: ID, roleName

Primary Key: ID

6. Users

Attributes: ID, username, password, roleId, phonenumber, gender, dob, identitycard, email

Primary Key: ID

Foreign Key: roleId references Role(ID)

Physical Design

The physical design involves the actual implementation of the database on a SQL Server, including data types, constraints, and storage details.

Data Types:

int for IDs and numeric fields.

varchar for text fields like bookname, ISBN, author, etc., with specified lengths.

datetime for date fields like yearofrelease, borrowFrom, borrowTo, and createdAt.

Constraints:

Primary keys are defined for each table to ensure unique identification of records.

Foreign keys are used to maintain referential integrity between related tables.

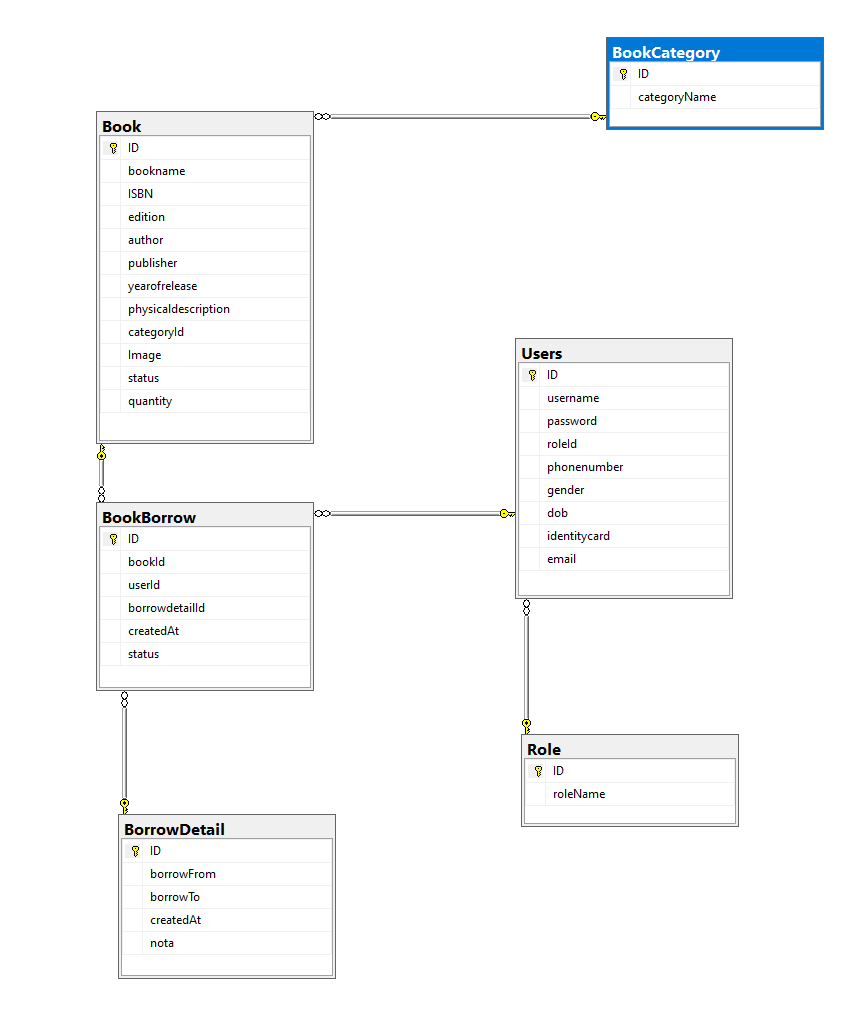
Clustered indexes are created on primary keys to optimize data retrieval.

Storage Details:

The database is created with a primary data file and a log file, with specified initial sizes and growth increments.

The database settings include options for auto-update statistics, recovery model, and query store configuration to enhance performance and reliability.

Database Diagram:



System Design

MVC Architecture

The Library Management System is designed using the Model-View-Controller (MVC) architecture, which separates the application into three interconnected components:

1. Model: Represents the data and the business logic of the application. In this system, the model classes include Book, BookBorrow, User, and BorrowDetail. These classes encapsulate the data attributes and provide methods to manipulate the data.

View: The user interface of the application. It is responsible for displaying data to the user and capturing user input. In a web application, this typically involves JSP pages or HTML files that render the data provided by the controller.

3. Controller: Manages the flow of the application. It processes user input, interacts with the model, and selects the view for response. In this system, servlets like BorrowManagementServlet and BorrowCheckoutServlet act as controllers.

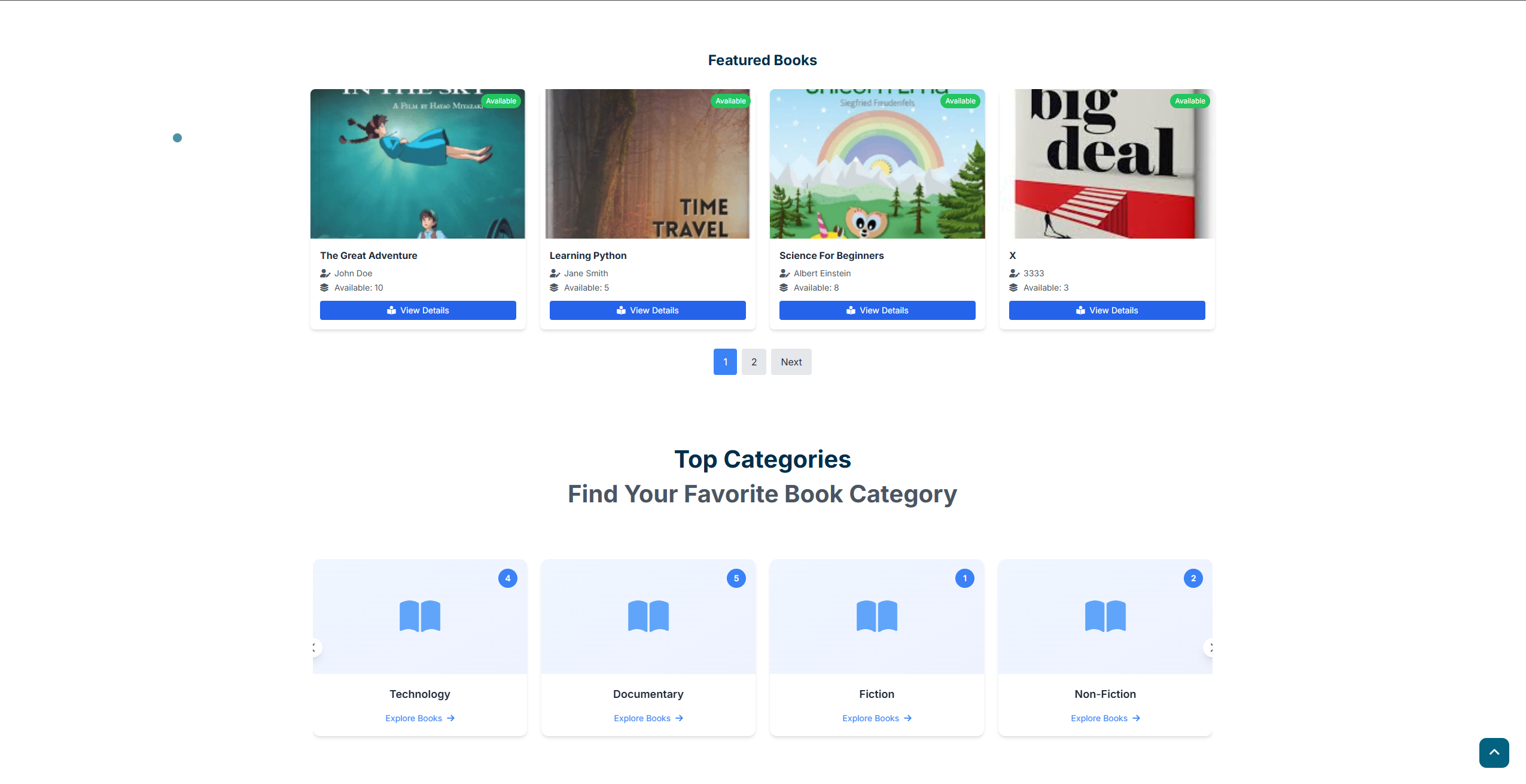
Implementation Details

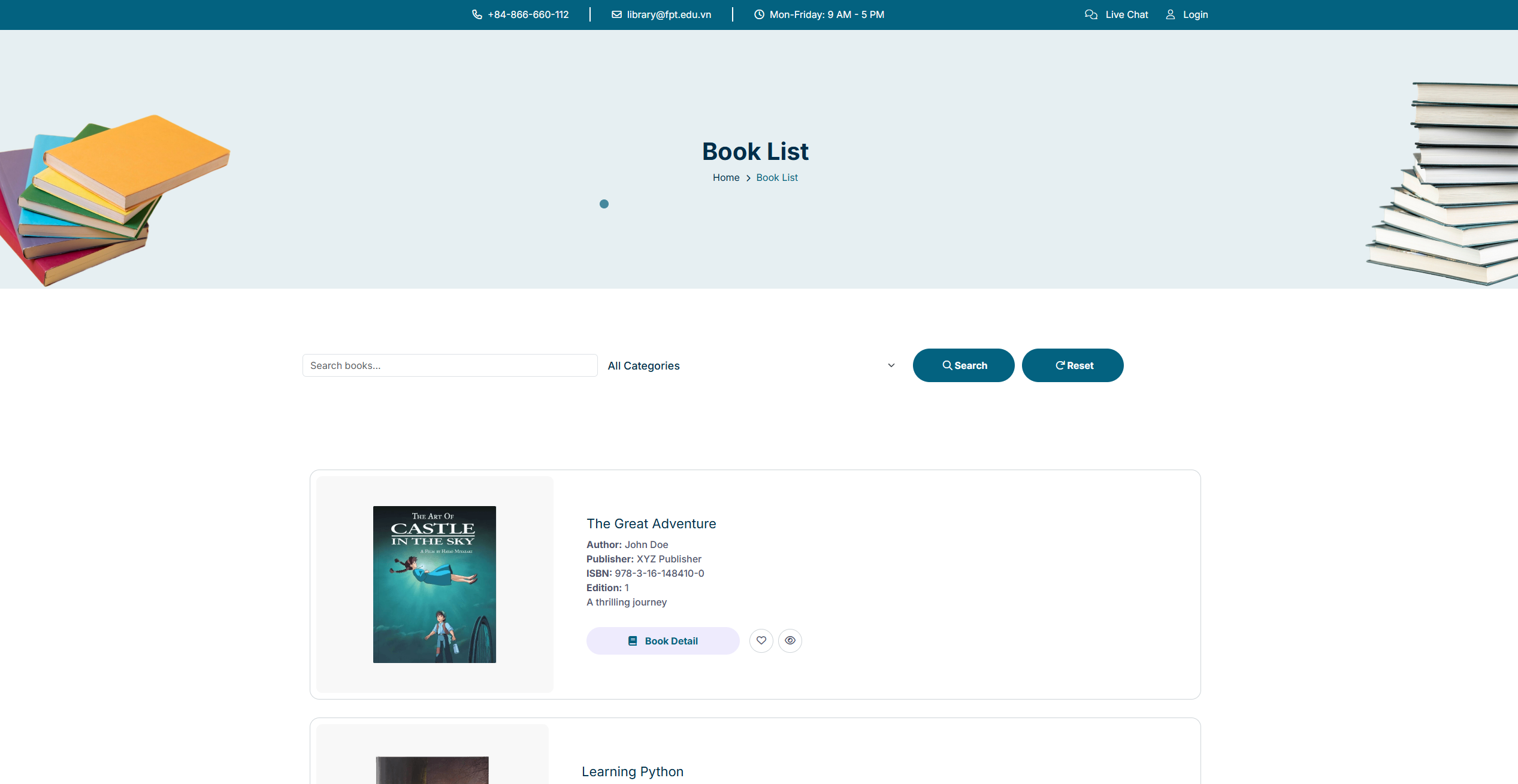
Data Access Layer (DAL): This layer is responsible for interacting with the database. Classes like DBContext, BookDAO, BookBorrowDAO, UserDAO, and BorrowDetailDAO handle database operations such as CRUD operations and queries.

Business Logic Layer: This layer contains the core functionality of the application, such as managing book borrowing, user registration, and book management. The business logic is implemented within the DAO classes and the model classes.

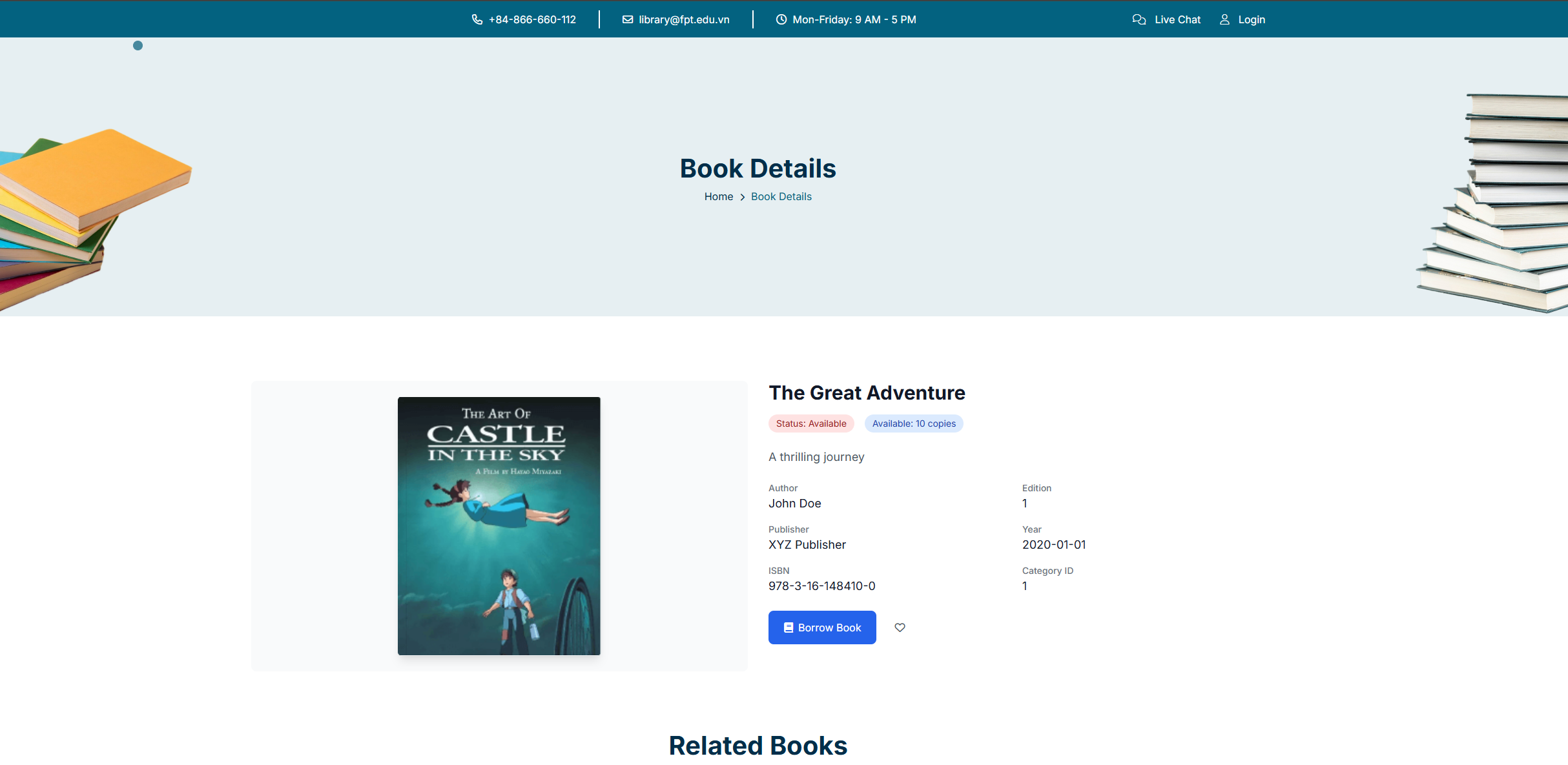
Presentation Layer: This layer includes the JSP pages and servlets that form the user interface. The servlets handle HTTP requests, interact with the business logic layer, and forward the results to the appropriate JSP pages for rendering.

4. ScreenShoot:

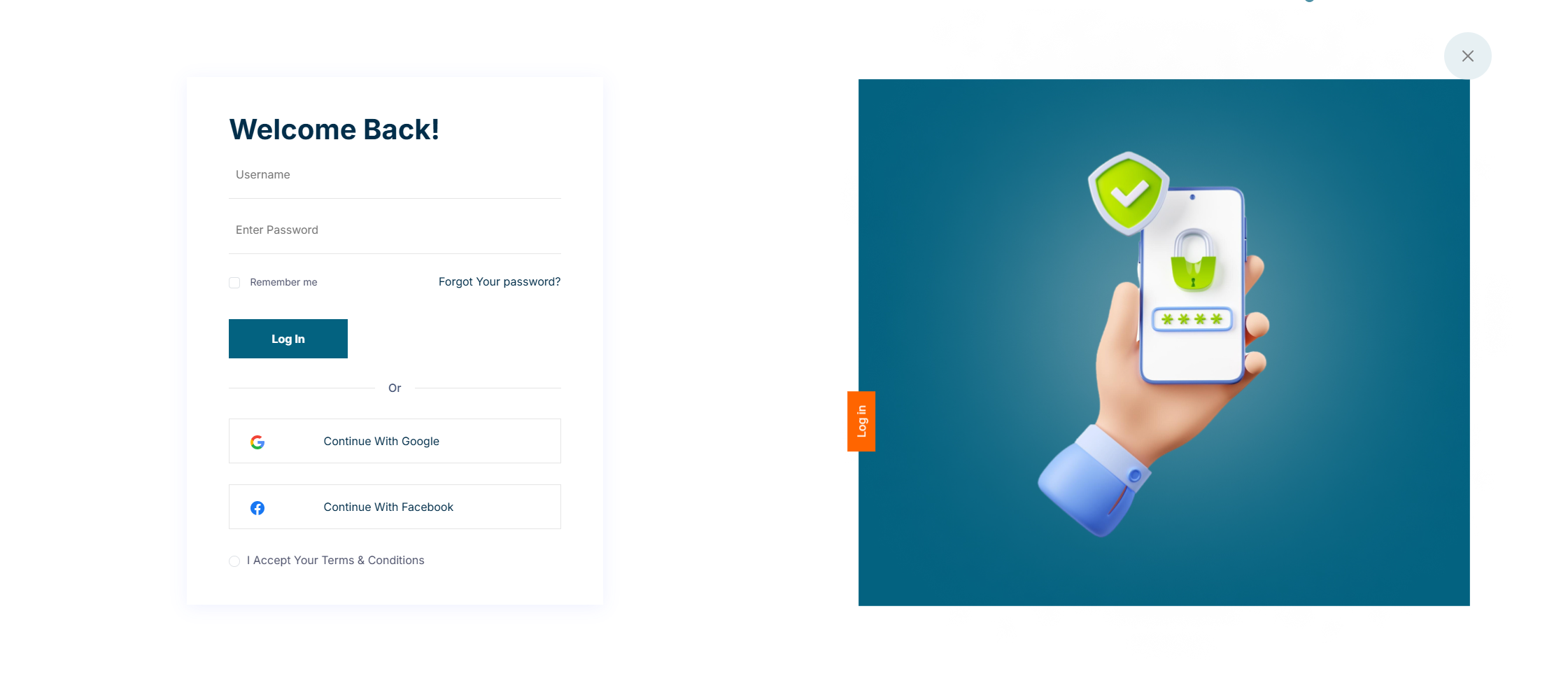
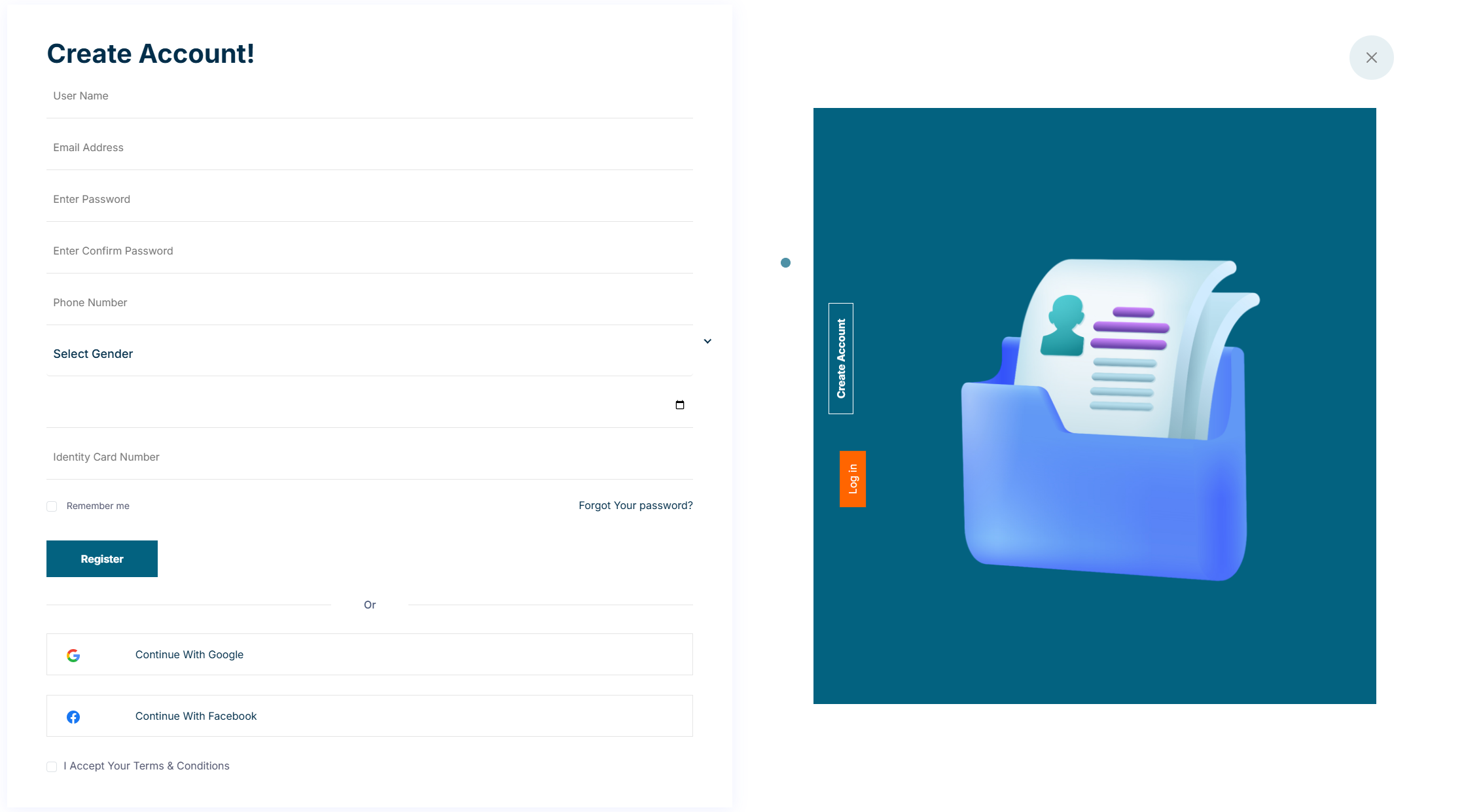
HomePage: Show featured Book and category



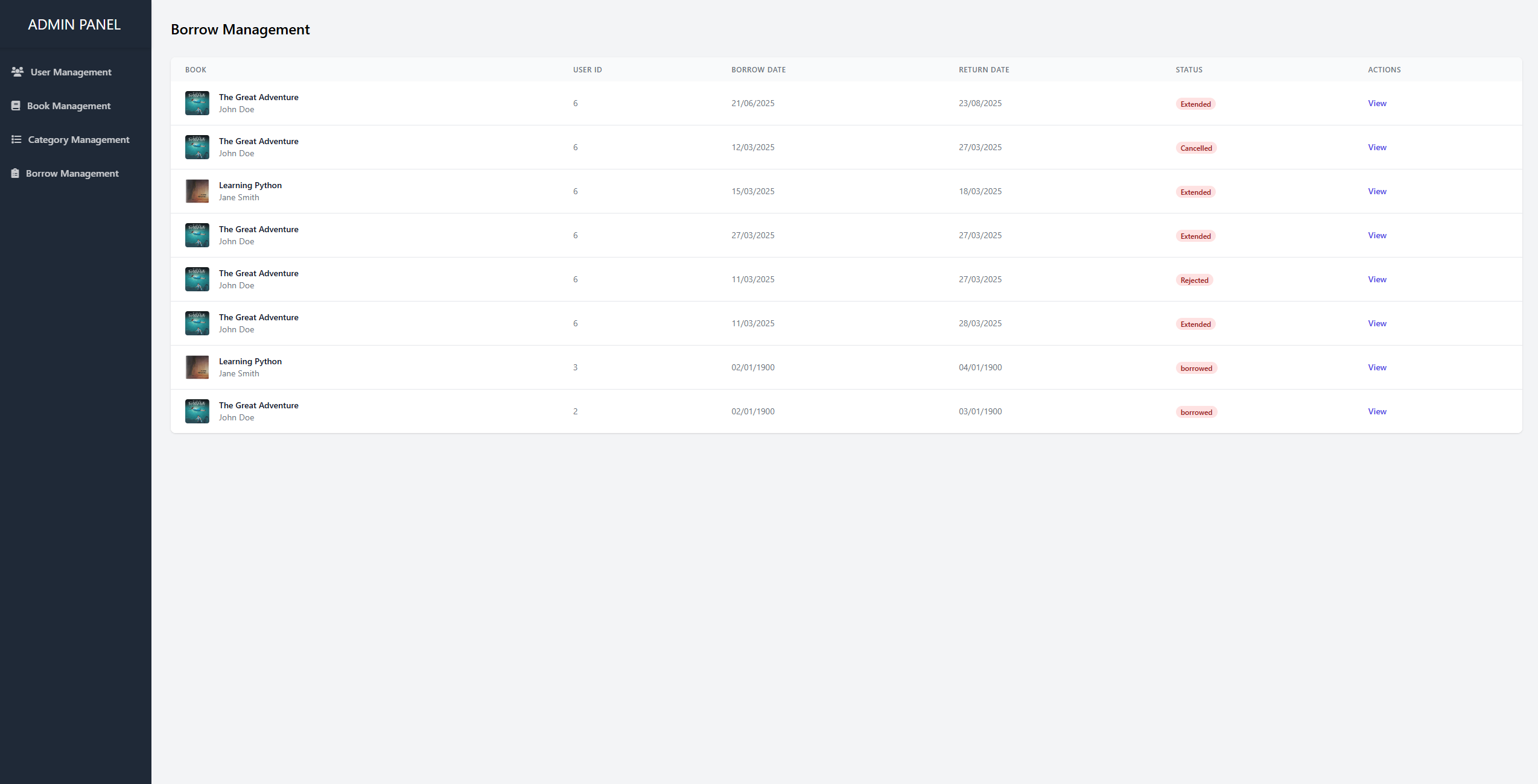
Book List: Show book list and filter book



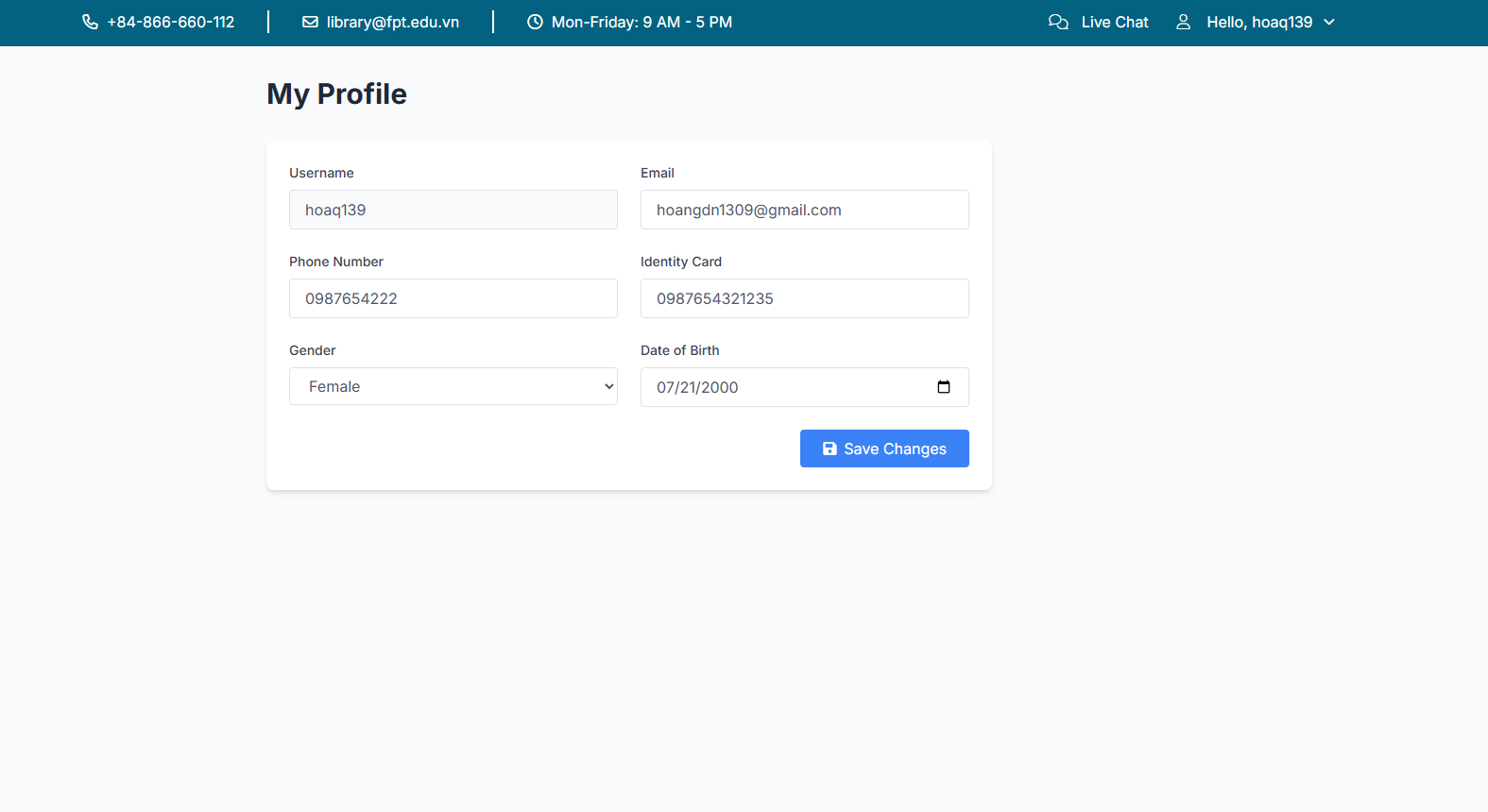
Show book details and related book

****

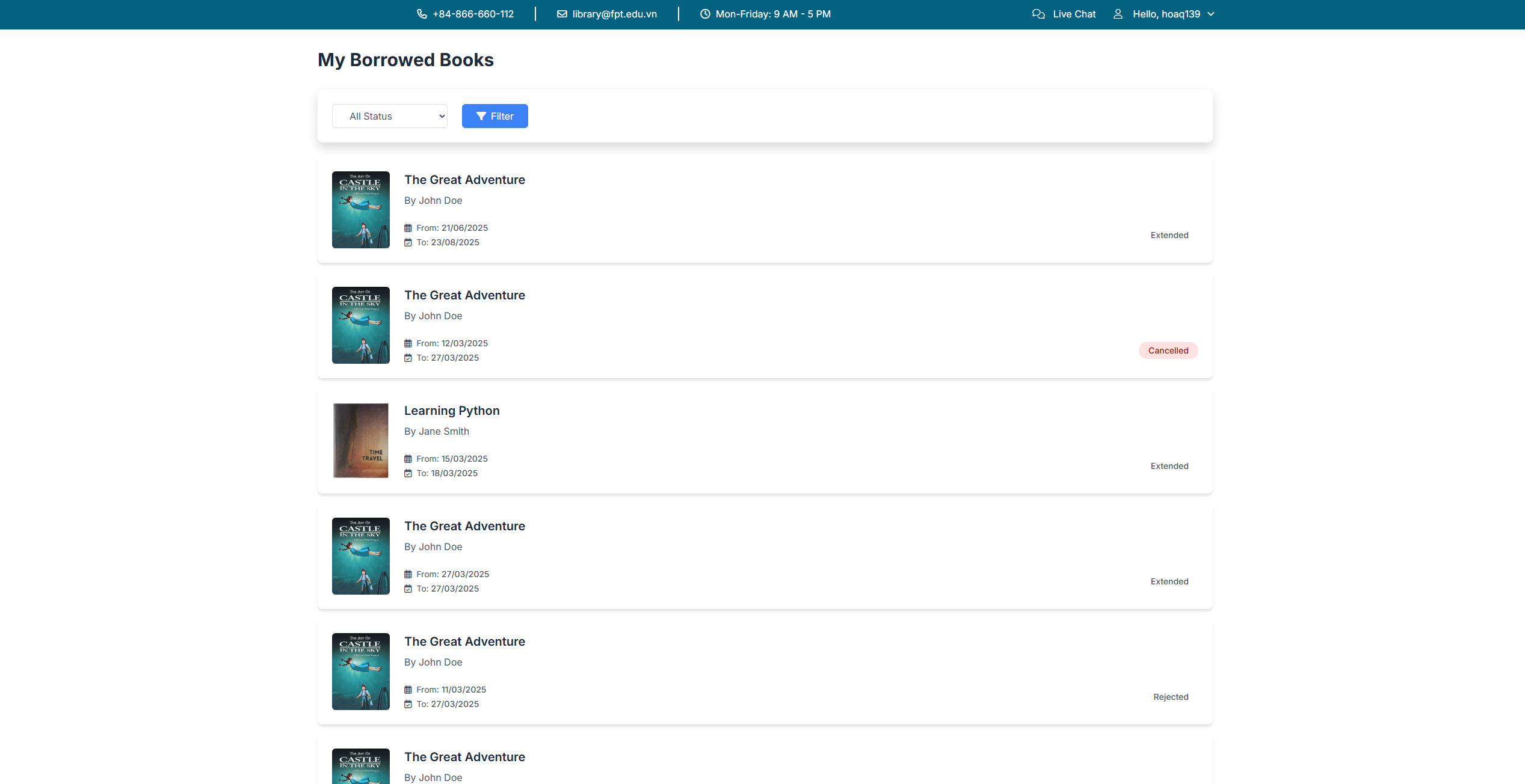
Login and Register: login and user register



Admin console: manage all the system



Manage user profile



Manage borrow history

6.

Pros of the Application

1. Modular Design: The application is structured using the MVC architecture, which separates concerns and makes the codebase more maintainable. This separation allows for easier updates and debugging.

2. Comprehensive Functionality: The application covers essential library management features, such as book borrowing, user management, and status updates, providing a robust solution for library operations.

Database Interaction: The use of JDBC for database operations ensures that the application can efficiently interact with the SQL Server database, allowing for reliable data storage and retrieval.

4. User Authentication: The application includes user authentication mechanisms, ensuring that only authorized users can perform certain actions, which enhances security.

Cons of the Application

1. Error Handling: The current error handling is basic, primarily using printStackTrace(), which is not suitable for production environments. More robust error handling and logging mechanisms should be implemented.

Scalability: The application may face scalability issues as the number of users and books increases. Optimizations in database queries and application logic are necessary to handle larger datasets efficiently.

3. User Interface: The user interface is not discussed in detail, but it is crucial for user experience. Enhancements in UI/UX design could make the application more user-friendly.

4. Code Duplication: There is some code duplication, especially in the way database operations are handled across different DAOs. This could be improved by implementing a more generic data access strategy.

Lessons Learned

Importance of Design Patterns: Implementing the MVC pattern has reinforced the importance of design patterns in creating maintainable and scalable applications.

Database Management: Working with JDBC and SQL Server has provided valuable experience in managing database connections and executing SQL queries.

Error Handling: The need for robust error handling and logging has become apparent, highlighting areas for improvement in future projects.