



# **Requirements Specification**

**Project Name: *eBook loaning tool***

**Team Number 01**

[Team 01]	<b>Specification Document</b>	[08/05/2025]
-----------	-------------------------------	--------------

#### Document Information

<b>Project Name:</b>	eBook loaning tool	<b>Preparation Date:</b>	06/03/2025
<b>Prepared By:</b>	Team 01		
<b>Email / Phone:</b>			
<b>Document Version No:</b>	2.4	<b>Document Version Date:</b>	08/05/2025

#### Version History

Ver. No.	Ver. Date	Revised By	Description
1.0	23/03/2025	Yunyi Jiang	Initial Draft
1.2	23/03/2025	Avin Jayatilake	Initial FR & NFR
1.3	23/03/2025	Peilin Li	Add User Cases
1.4	24/03/2025	Guanyuan Wang	Hardware and software
1.5	25/03/2025	N.Amin	Assumption and changes in FR & NFR/Domain analysis
1.6	25/03/2025	Grace Brown	Analysis Process 1.1, Analysis 1.2, UML diagram
1.7	25/03/2025	Avin Jayatilake	Table of contents, updated FR & NFR, definition of terms
1.8	25/03/2025	Peilin Li	Constraints and Dependencies & Initial ideas for a GUI.
2.0	08/05/2025	Peilin Li	Add Scope, Updated Hardware and Software, Assumptions and Back End
2.1	08/05/2025	Guanyuan Wang	Add use case diagram
2.2	08/05/2025	Grace Brown	Updated FR&NFR, Definition of terms, GUI design
2.3	08/05/2025	Avin Jayatilake	Updated table of contents, FRs,NFRs, Purpose, Definition of terms, Background and Analysis, References
2.4	08/05/2025	Yunyi Jiang	Add UML and ER parts

## Table of Contents

Purpose (Executive Summary) .....	4
1. Background & Analysis.....	4
1.1 Analysis Process.....	4
1.2 Analysis.....	5
1.3 Scope.....	7
2. Hardware & Software to be used.....	8
2.1 Hardware .....	8
2.2 Software .....	9
3. References.....	10
4. Definition of terms.....	11
5. Solution requirements .....	13
5.1 Functional Requirements.....	13
5.2 Non-Functional Requirements.....	14
6. Other considerations.....	18
6.1 Assumptions.....	18
6.2 Constraints & Dependencies .....	19
6.2.1 Constraints.....	19
6.2.2 Dependencies.....	19
7. Initial requirements analysis of user interaction.....	19
7.1.1 UML Activity diagram .....	20
7.1.2 UML Class Diagram.....	21
7.1.3 UML Package Diagram .....	21
7.1.4 ER Diagram.....	22
7.2 Initial ideas for GUI.....	23
<b>7.2.1: Version 1.0 of the Design.....</b>	<b>23</b>
<b>7.2.1: Version 2.0 of the Design.....</b>	<b>25</b>
7.3 Use cases.....	33
8. Back End .....	37

[Team 01]	<b>Specification Document</b>	[08/05/2025]
-----------	-------------------------------	--------------

## Purpose (Executive summary)

The purpose of this document is to detail the requirements that need to be satisfied to achieve the project's objective. This requirement specification document ensures that all team members have a shared understanding of the project goals, guaranteeing that all necessary in-scope requirements are met while avoiding out-of-scope topics. If all these requirements are fulfilled, the project will be considered successful.

The project brief our team received involves developing an "eBook loaning system" that allows users to borrow eBooks from a centralised database. The application is designed to be a website. The basic requirements include user account management, eBook borrowing and return management, loan history tracking, a book review system, and email notifications. The system should also provide cover images, categories, and descriptions for eBooks to assist users in making informed choices.

There are benefits of the system for the two groups of stakeholders. The first of these stakeholders are regular users of the system: users can browse the eBook catalog, create and manage accounts, and borrow up to 10 eBooks simultaneously. They can also create a wish list, cancel loans, update personal information, and leave reviews on borrowed eBooks. Once the loan period expires, the eBook will be automatically returned without any fines.

The second stakeholder is the admin: Admins can manage the eBook inventory, add, remove, and edit eBook availability and loan duration, and monitor user operations and user reviews.

The objectives of this system are to satisfy at a minimum, the basic requirements in the specification sent to us. These basic requirements can be referred to in an earlier paragraph. In addition to meeting the basic requirements, the team must also adhere to the assumptions set out by the specification. The assumptions are that each user can leave multiple comments on the same book. These requirements will be discussed in further detail later in this document in sections 5.1 (functional requirements) and section 5.2 (non – functional requirements).

A secondary objective for our team will be to consider some possible extensions which would be useful if they are deemed suitable. These will be discussed in section 1.3 (scope), where distinct boundaries about what the team should and should not include in the system are established.

The team's goal is to ensure that all basic requirements outlined in the project brief are met and that all stated assumptions are adhered to before the deadline 9th May 2025. The system will undergo rigorous testing to ensure functionality and stability.

A list of technical terms used in this document is available in section 4.0 of this document for reference.

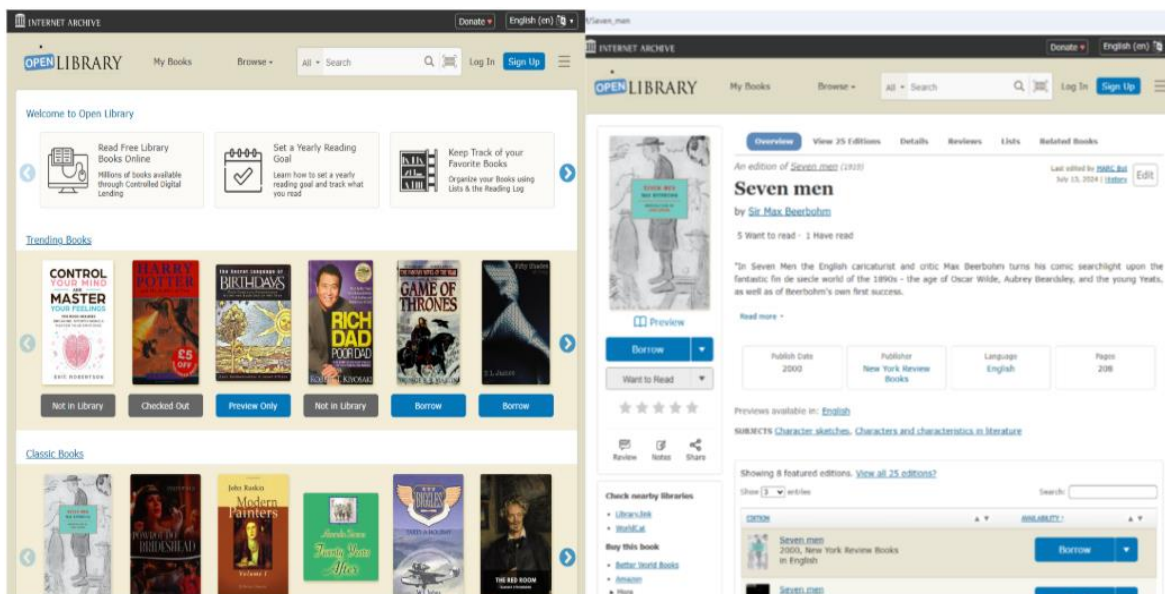
## 1. Background & Analysis

### 1.1. Analysis Process

Creating a web base application by which public user can explore eBooks and have a look on the details which included the title, cover page, review and comments without registration requirements.

The team was tasked to develop the system that allows users with an account to loan eBooks from a centralized database. The basic functionalities include allowing users to browse and borrow eBooks, manage their loans through their account dashboard, receive notifications via email for loan confirmations, cancellations, and expirations and finally, users should be able to leave reviews on books via 5-star rating system.

To ensure the app met expectations, market research on various eBook loaning systems was conducted to identify the key functionalities that the team asked to, and what could be improved on. The team focused on generally on eBook loaning platforms such as Amazon's Kindle Library Lending, Newcastle University Library, and Google Play Books. However, the team found the closest eBook library which satisfied project requirement features was OpenLibrary. After carefully evaluating the strengths and weaknesses of each system, the team identified several aspects they wanted to incorporate. Each key requirement was researched, and the best features of each system was used to develop the design of the app simplify for UI/UX purposes.



### 1.2. Analysis

#### Existing Systems and Processes

Existing systems and processes were analyzed to understand the way eBook loan systems work and interact with users. For this task, the team needed to conduct market research that investigated the pros and cons of current systems. This was completed by looking at each requirement specified in the project brief and attempting to complete this requirement on each system. This process allowed the

team to gain a deep understanding of how eBook loaning systems operate, and what an effective eBook loaning systems looks like. It also helped the team locate areas of improvement and poor user design.

As previously mentioned, these systems were used during this research: Amazon's Kindle Library Lending, Newcastle University Library, and Google Play Books and OpenLibrary.

To look at the effectiveness of each system, the team went through each requirement and tested it:

1. Can you create and manage your account? What are the pros/cons?
2. What is the process for searching for a ebook?
3. What is the process for borrowing a ebook?
4. How do you view/add to your wish list?
5. How do you write a review? Is there a certain rating system that is used?

### Observations

After careful analysis, the following observations were taken into consideration:

- The login button needs to be clearly visible- This is going to be the first thing the user looks for
- Sign in details should not be over complicated: Name, Email, Password
- A dashboard is extremely handy for a user to manage their account. The dashboard should include loan history, loan management, security (change password), notifications/messages.
- When searching for a book, there should be a clear search bar
- When browsing, books should have a clear image, title, author and brief description.
- After a loan has started, users should be able to clearly clarify the duration of the loan, the end date of the loan.
- The Wishlist should include the image and name of the book so it can be found easily
- When searching for books, a button needs to be accessible so the book can be added to the Wishlist with ease
- Email reminders for loan confirmation, cancellations, and expiries
- Reviews should be accessible when viewing the book, usually at the bottom after the information about the book
- A 5-star rating system is straight forward and easy to follow
- The number of reviews shows a books popularity- the more reviews the more users are reading it

### Stakeholders

The main stakeholders are users of the system, those who are searching and borrowing the eBooks. As they are the primary stakeholders, it is integral that the system is straight forward and easy to use so that they are not discouraged from using the system. The second stakeholder will be admin users who are responsible for maintaining and offering eBook loaning service to the users. Admins can manage the eBook inventory, add, remove, and adjust eBook availability and loan duration, and monitor users and posted reviews. This is an equally important role; without the admin the system will fail to work. The third stakeholders are public users who don't want or have account in the system they will be able to explore and see books with relevant details but will not be able to borrow, leave comments and reviews.

The market research for this task is an important step in designing the system and ensuring its success. A clear and user-friendly interface will improve usability and enhance the user experience.

### 1.3. Scope

## In Scope

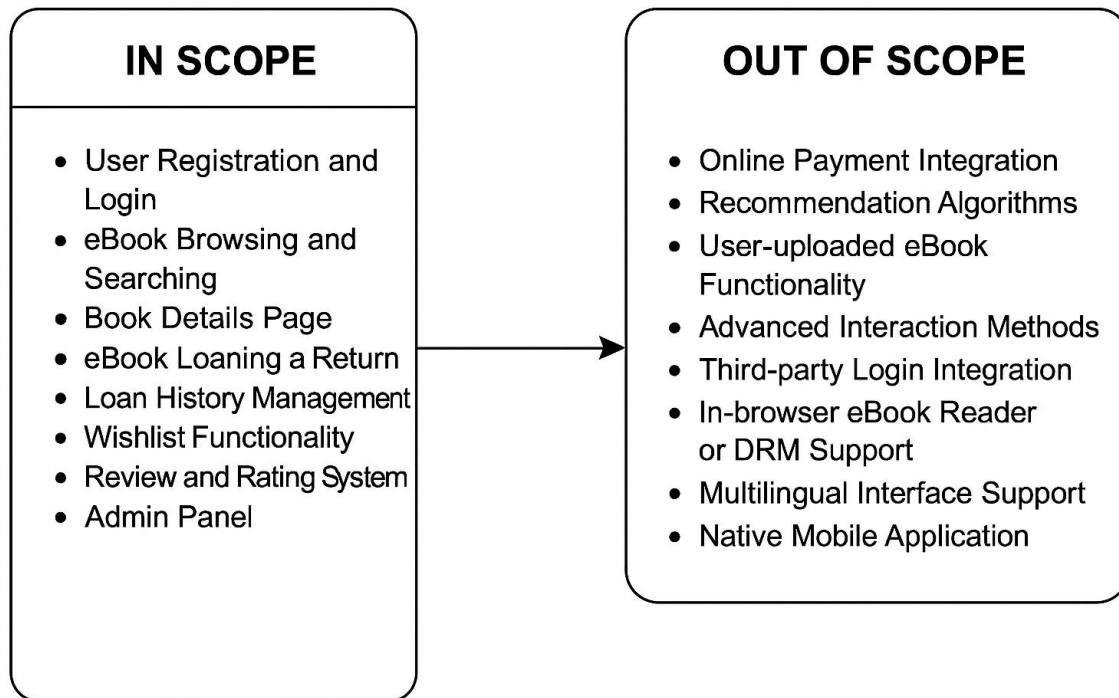
The project aims to develop a user-friendly and fully functional web-based eBook loaning platform. The following features are explicitly included within the scope of development:

- **User Registration and Login:** Including email-based verification and authentication.
- **eBook Browsing and Searching:** Users can filter by category or search using keywords.
- **Book Details Page:** Displays cover image, title, author, description, availability, and reviews.
- **eBook Loaning and Return:** Users can borrow up to 10 books, with automatic return upon expiry.
- **Loan History Management:** Users can view a history of borrowed eBooks.
- **Wishlist Functionality:** Add and remove books from a personal wishlist.
- **Review and Rating System:** Users can leave reviews and rate books after borrowing;
- **Admin Panel:**
  - Add, edit, and delete eBooks.
  - Manage eBook availability and loan durations.
- **Email Notification System:**
  - Notifications for registration, loan confirmation, cancellation, and expiry reminders.
- **User Settings:** Ability to update personal information, reset password, or delete account.
- **Responsive Web Design:** Fully accessible across desktop and mobile devices.

## Out of Scope

To ensure timely delivery of the project, the following features are explicitly excluded from the current scope, though they may be considered for future development:

- **Online Payment Integration** (e.g., rental fees, late fees).
- **Recommendation Algorithms** for personalized book suggestions.
- **User-uploaded eBook functionality.**
- **Advanced interaction methods** (e.g., voice search, OCR-based scanning).
- **Third-party login integration** (e.g., Google, Facebook).
- **In-browser eBook Reader or DRM support** (the system handles borrowing only, not reading).
- **Multilingual interface support** (the application will be English-only).
- **Native Mobile Application Development** (only a web-based version is included).



## 2. Hardware and software platforms to be used for developing and running the solution

### 2.1 Hardware

Although the project does not have mandatory hardware requirements, to ensure system compatibility and responsiveness, development and testing were conducted on the following devices:

Device Model	Operating System	Processor	Memory	Purpose
Lenovo Legion Y7000	Windows 11 Pro	Intel Core i7-9750H	16 GB	Backend development and testing
MacBook Air M2	macOS Sonoma 14.3	Apple M2 Chip	8 GB	Frontend development and UI design

All devices accessed the system via a stable broadband connection, and cross-platform functionality was tested on multiple modern browsers.

#### Recommended Minimum Hardware Requirements:

- Processor:** Intel Pentium 4 or higher
- Memory:** 128MB (≥1GB recommended)
- Resolution:** Minimum 1024×768



[Team 01]	<b>Specification Document</b>	[08/05/2025]
-----------	-------------------------------	--------------

- **Network:** Stable internet connection

## 2.2 Software

The system was developed using a modern full-stack technology toolkit, covering frontend, backend, database, testing, and collaboration control. The specific software and their versions are listed below:

### Frontend

- HTML5, CSS3, JavaScript (ES6+)
- **Tailwind CSS v3.4.1:** Used for responsive design
- **Browser Testing Environment:**
  - Google Chrome v123
  - Mozilla Firefox v125
  - Microsoft Edge v123
- **Figma:** Used for UI prototyping

### Backend

- **Java JDK 17.0.9**
- **Spring Boot v3.2.1:** For building RESTful API services
- **Maven v3.9.9:** For project building and dependency management

### Database

- **MySQL v8.0.36:** Stores data such as users, books, and borrow records
- **DBeaver v23.3:** GUI-based database management tool
- **JDBC Driver v8.0.33:** Backend-to-database connectivity interface

### Testing & Collaboration Tools

- **Postman v10.23:** For testing API functionality
- **JUnit v5.10:** Backend unit testing framework
- **Git v2.42.0 + GitHub:** For version control and team collaboration
- **IntelliJ IDEA v2023.3.2 / VS Code v1.88.0:** Integrated development environments (IDEs)

## 3. References

**Amazon**, 2025. *Kindle Store*. [online] Available at: <https://www.amazon.co.uk/Kindle-eBooks> [Accessed 8 Mar. 2025].

**Chen, J. and Pan, H.**, 2020. 'Design of Man Hour Management Information System on SpringBoot Framework', *Journal of Physics: Conference Series*, 1646(1).

**De, B.**, 2023. *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization, 2nd Edition*. 2nd ed. Place of publication not identified: Apress.

**Google**, 2025. *Google Play Books*. [online] Available at: <https://play.google.com/store/books> [Accessed 8 Mar. 2025].

**Harrand, N., Benelallam, A., Soto-Valero, C., Bettega, F., Barais, O. and Baudry, B.**, 2022. 'API beauty is in the eye of the clients: 2.2 million Maven dependencies reveal the spectrum of client-API usages', *The Journal of Systems and Software*, 184, p.111134.

**Internet Archive**, 2025. *OpenLibrary*. [online] Available at: <https://openlibrary.org> [Accessed 8 Mar. 2025].

**Newcastle University**, 2025. *Library*. [online] Available at: <https://www.ncl.ac.uk/library> [Accessed 8 Mar. 2025].

**Nielsen, J.**, 1994 (updated 30 Jan. 2024). *10 Usability Heuristics for User Interface Design*. [online] Available at: <https://www.nngroup.com/articles/ten-usability-heuristics/> [Accessed 1 May 2024].

**Späth, P.**, 2023. 'Corporate Maven Repositories', in *Pro Jakarta EE 10*. [online] Berkeley, CA: Apress, pp. 61–86.

**Streib, J.T., Soma, T. and SpringerLink**, 2023. *Guide to Java: A Concise Introduction to Programming*. 2nd ed.

#### 4. Definition of terms

**Apache Tomcat**: A free and open-source web server that provides a "pure Java" HTTP server environment for running Java applications.

**Bootstrap**: A popular front-end framework for developing responsive and mobile-first websites using HTML, CSS, and JavaScript.

**CSS3**: The latest version of Cascading Style Sheets, used for styling web pages with enhanced features like animations and media queries.

**Cypress**: Modern end-to-end testing framework for web applications.

**DBeaver**: A free, open-source database management tool that supports multiple database types, including MySQL,

**Figma**: web-based design and prototyping tool used for user interface and user experience design

**GB (Gigabyte):** A unit of digital storage equal to **1,024 MB**.

**Git:** A version control system that tracks changes in code, enabling developers to collaborate and manage code efficiently.

**GitHub:** A web-based platform for version control and code collaboration, built around Git.

**Hardware:** The physical components of a computer system, such as the processor, memory, and storage devices.

**HTML5:** The latest version of Hypertext Markup Language, used for structuring web pages and supporting multimedia without additional plugins.

**IDEs (Integrated Development Environments):** Software applications that provide tools for coding, debugging, and testing, such as IntelliJ IDEA and Visual Studio Code.

**Intel Pentium 4 Processor:** A single-core processor developed by Intel, commonly used in computers during the early 2000s.

**IntelliJ IDEA:** A popular IDE for Java development that offers advanced coding assistance, debugging, and integration with frameworks like Spring Boot.

**Java:** A versatile, object-oriented programming language widely used for building web, desktop, and mobile applications.

**JavaScript:** A high-level programming language used for creating dynamic and interactive web pages.

**JDBC (Java Database Connectivity):** A Java API that allows applications to connect to and interact with relational databases.

**JDK 17 (Java Development Kit 17):** A development kit that provides tools and libraries for building and running Java applications.

**Maven:** A build automation tool used primarily for Java projects, managing dependencies, project structure, and build processes.

**MB (Megabyte):** A unit of digital storage equal to **1,024 KB**.

**MySQL:** An open-source relational database management system used for managing structured data.

**Postman:** A tool for testing REST APIs, allowing developers to send requests, inspect responses, and automate API testing.

**RAM (Random Access Memory):** A type of computer memory that stores data temporarily while programs are running, improving system performance.

**Resolution:** The clarity and sharpness of a display, defined by the number of pixels on the screen (e.g., **1920x1080**).

**REST APIs (Representational State Transfer Application Programming Interfaces):** Web services that follow REST principles, enabling communication between client and server using standard HTTP methods.

**Software:** Programs and applications that run on computer hardware to perform specific tasks.

**Spring Boot:** A Java framework designed to simplify the development of web applications and microservices by providing pre-configured settings and built-in tools.

**Tailwind CSS:** A utility-first CSS framework that enables developers to create responsive designs directly within their HTML.

**Visual Studio Code:** A lightweight, open-source code editor developed by Microsoft, supporting multiple programming languages and extensions.

**Vue:** A progressive JavaScript framework for building user interfaces, focusing on flexibility and simplicity.

**W3C Markup Validation Service:** This tool checks your HTML or XHTML code to ensure it follows W3C standards.

**W3C CSS Validation Service:** This tool checks your CSS code for syntax errors and compliance with W3C CSS specifications.

## 5. Solution requirements

For each requirement indicate its Priority e.g. High, Medium, Low.

### 5.1. Functional Requirements

Requirement	Priority (H, M, L)	Supplier Comments
FR0 – The system shall provide a login page that allows users to enter a username and password	H	Users must have an account to borrow books and add to Wishlist, view user details, and update email and password
FR1 – Admin will have specific user credentials to login	H	Admin will have access to user and admin pages
FR2- User will receive an email when registration has been successful	H	
FR3 – Any user will be able to view an eBook cover image, title, author and description	H	
FR4 – Any user will be able to filter by genre and search by keywords	H	
FR5 – Admin should be able to edit and delete eBooks.	H	
FR6 - Admin should be able to search books by title	H	
FR7 - Admin should be able to search registered users	H	
FR8 - Admin should be able to upload new books	H	
FR9 – Admin should be able to manage reviews	H	
FR10- The system shall store a user's ID, name, email and password	H	
FR11- Users can update their email and password once logged in	H	
FR12 - Users can add eBooks to their Wishlist and view all books in the Wishlist	H	

[Team 01]	<b>Specification Document</b>	[08/05/2025]
-----------	-------------------------------	--------------

FR13- Users can borrow eBooks for 14 days, and return them earlier if needed	H	
FR14 – Users shall receive an email when they borrow or return an eBook	M	
FR15 – Users shall receive a reminder email 3 days before expiration date.	H	
FR16- Users can view all books currently on loan in my loans	H	Books will disappear once loan is over
FR17- Users shall be able to leave reviews on borrowed eBooks	H	User must have loaned a book first to write a review

## 5.2. Non-Functional Requirements

Requirement	Priority (H, M, L)	Supplier Comments
<p>NFR0- Users will need the following information to login or register an account:</p> <ul style="list-style-type: none"> <li>• Username</li> <li>• Email</li> <li>• Password (twice to ensure it is correct)</li> </ul> <p>Passwords must match for registration</p> <p>User cannot use the same email twice</p> <p>All input fields must be filled in</p> <p>The Spring Boot REST API will validate response and add user to the database during registration. During login, the Spring Boot API will very user credentials against existing credentials already in the database</p> <p>Passwords will be securely stored using hashing. They should be encrypted so that there is no unauthorized access.</p> <p>Pop-up window will be received when registration is successful</p> <p>Redirects user to Home Page once login is complete</p>	H	Essential requirement for users to borrow from the eBook library. Passwords with a hashing mechanism ensures password integrity
NFR1- Admin details will be stored on the database	H	Admin will be given specific login details when becoming admin. This can be accessed on the README file
NFR2 – A confirmation email should be sent to the user within 1 minute of successful user registration	H	Ensures the user knows straight away if registration has been successful

<p>NFR3 - Users can update their password in the Login &amp; Security page. They will need to enter their current password and retype their new password twice.</p> <p>Once the details have been correctly sent, a Spring Boot REST POST request will be sent to change the user's password</p> <p>A pop-up window will say the password has been changed successfully</p>	H	The user will be notified if passwords do not match. Ensures they do not accidentally create a wrong password
<p>NFR4- Users will be able to view their email and username, as well as change their email in the My Details page.</p> <p>GET request from the Spring Boot REST API will be used to retrieve a user's email and username</p> <p>POST request from the Spring Boot REST API will be used to change the user's email</p> <p>Pop up window when change of email has been successful</p>	H	
<p>NFR5 – Only the admin should have access to the book management features. Book modifications (edit or delete) should be reflected in the system within 5 seconds. The admin book management features should include intuitive interfaces for even a non-technical admin user to manage the book catalogue efficiently through provision of simple and clear buttons</p>	H	In the case of editing book details such as the title, author, genre or description, the admin can select the 'Edit' button. The admin can delete book by selecting the 'Delete' button.
<p>NFR6 – The admin can search for books in the search bar provided in the Book Management page</p>	H	
<p>NFR7 – The admin can search for users through typing usernames in the search bar in the User Management page. Additionally, the admin can select the numbered page to load the next list of registered users</p>	H	The admin can see registered users with their corresponding email address they used for registering. To delete users, the admin can select the 'Delete' button. To see more users the admin can select the page button number (1,2,3,4 or 5).
<p>NFR8- To upload a new book the admin can enter the relevant book details in the Upload Books section.</p>	H	Admin is presented with a confirmation message "Are sure you want to delete this user" before deleting a user.

NFR9 – The admin can see which user has written a review for a book and. Statuses of reviews will be updated with colored wordings for clarification purposes	H	Once a user has posted a review their review status will be pending until the admin takes action on the review. The admin can either approve or dismiss the review. A confirmation message will appear before approving or dismissing a review. Once a review is approved the status will be “Approved” in green, or if the review is dismissed the status will be “Dismissed” in red.
NFR10- Books can be added to the Wishlist in the book details section  POST Spring Boot REST API request will add book into the Wishlist page	H	Book Details can be read and button to add to the Wishlist straight away after reading the details of the book
NFR11 – The Home Page should load within 2 seconds displaying a navigation bar and a selection of eBooks with corresponding details  GET Spring Boot REST API request to retrieve all books from the database.  The layout should be clear, with book details easily visible and organized. The system should support multiple users browsing the catalogue without degrading the performance.  The interface should follow consistent design patterns to improve user experience.,	H	
NFR12 – Search results should be returned within 2 seconds. Users should be able to access key features, such as book searching, with a few numbers of clicks.  Users must press enter on their keyboard to see results	H	
NFR13 – The system will allow maximum 10 eBooks to be borrowed by a single user. The eBooks system will not allow to exceed the number of available eBooks.	H	
NFR14 – An email will be immediately sent out reminding users the length of time an eBook can be loaned. Reminder emails should be sent 3 days before expiration and delivered within 30 seconds. The system should maintain loan expiry data with 100% accuracy to ensure reminders are timely.	H	Users may have to check their spam emails

NFR15 – Books can be loaned using the borrow button in either the Book Details section or the Wishlist section.  POST Spring Boot REST API request will add book into the my loans section	H	
NFR16- Once a user has loaned a book, the date it was borrowed, when it is due and the status will be displayed in the My Loans page  GET Spring Boot REST API used to receive borrow/return details from the database	H	
NFR17 – Users reviews will be approved or dismissed through the admin. This ensures a reliable user interaction with the system. A clear text box will be provided for reviews to be submitted. Users can also select the rating. The user cannot submit the review until a review is typed, and a star rating is selected	H	To select the rating the user can click the drop down 'Select a rating' button and choose from a scale of 1-5 stars. Once the user has clicked 'Submit Review' a confirmation message will appear notifying the review has been successfully posted.
The system should satisfy Nielsen's 8 <sup>th</sup> heuristic of Aesthetic and Minimalist design (Nielsen, 1994). This will be achieved by ensuring a consistent design is maintained throughout the various pages and only the essential information relating about the functional requirements are displayed	M	
The system should satisfy Nielsen's 5 <sup>th</sup> heuristic of Error Prevention (Nielsen, 1994). This will be accomplished by thorough various unit and user testing to eliminate errors. Also the system will provide useful feedback messages for users and the admin when error handling is experienced.		

## 6. Other considerations

### 6.1. Assumptions

- Each person can be associated with only one user account.
- Browsing the eBook catalog is allowed without requiring user login or account creation.
- Users can only borrow eBooks if there are available copies; the system will prevent loans from exceeding availability.
- If an eBook loan exceeds its allowed duration, it will be automatically removed from the user's loan list, and no overdue fines will be applied.

### 6.2. Constraints and Dependencies

#### 6.2.1 Constraints



- Users must have a valid internet connection to access and interact with the system.
- System access for certain functionalities (e.g., borrowing, reviewing, wish listing) requires user authentication.
- The backend must comply with standard RESTful API practices to ensure compatibility and scalability.
- System performance may vary depending on the user's device and network stability.

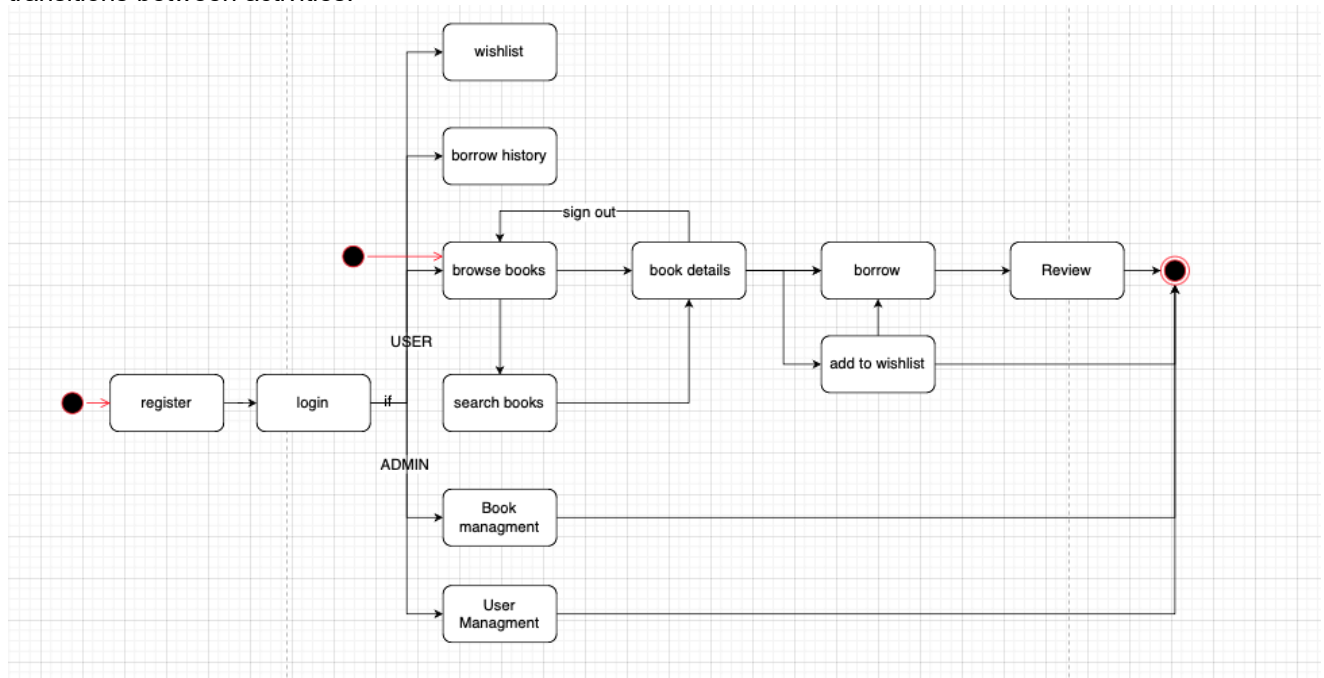
#### 6.2.2 Dependencies

- User authentication and session management depend on proper integration with the backend (Spring Boot + JWT).
- Database connectivity relies on a stable MySQL service and properly configured JDBC drivers.
- Frontend responsiveness depends on browser compatibility and CSS framework (Tailwind CSS) support.
- Collaborative features and version control depend on GitHub's availability and Git integration.

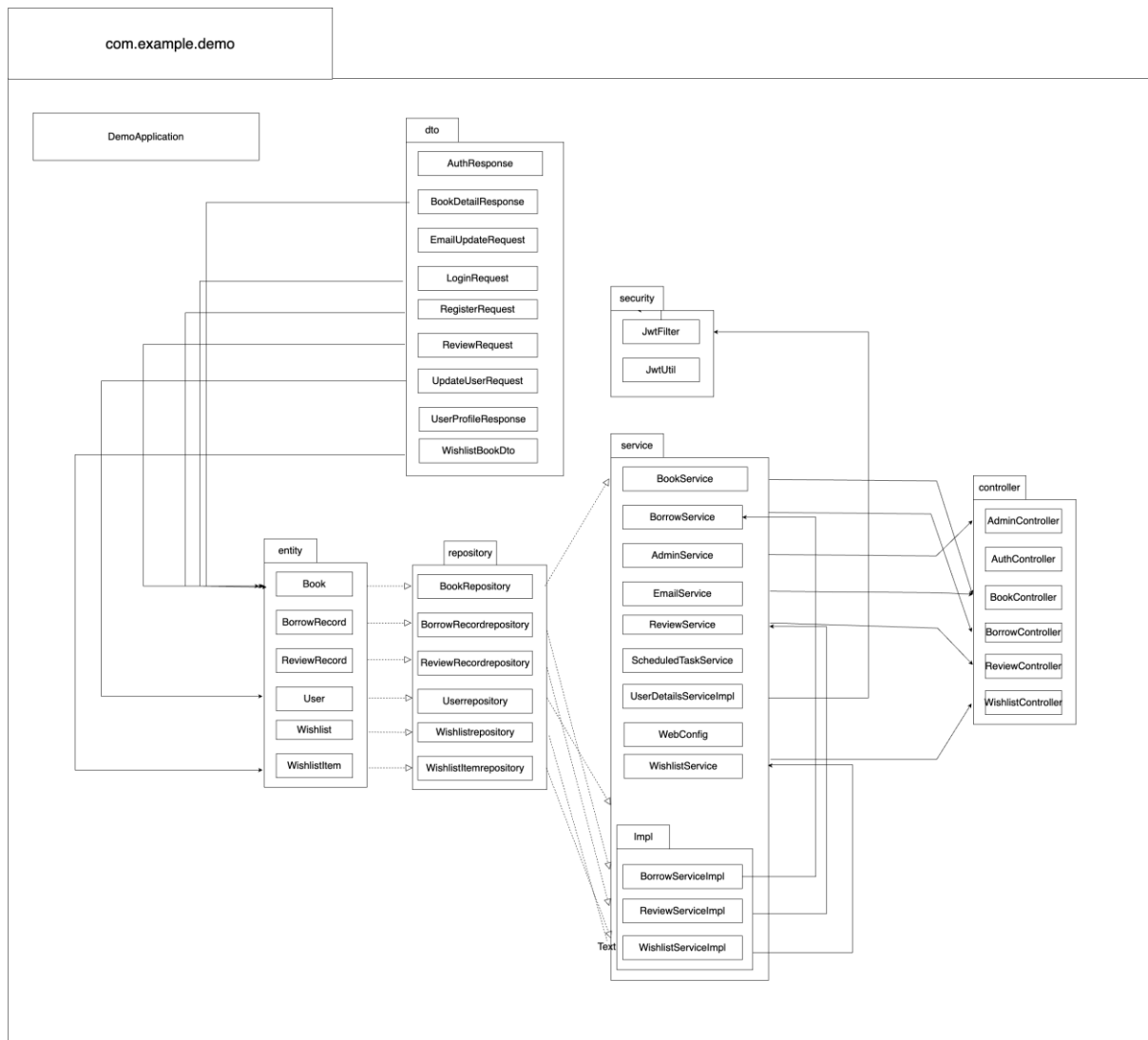
## 7. Initial requirements analysis of user interaction

### 7.1.1 UML Activity diagram

Activity diagram - This diagram shows the user behaviour for placing an order. Each activity within this process is in a rounded rectangle, connected by arrows showing transitions between activities.

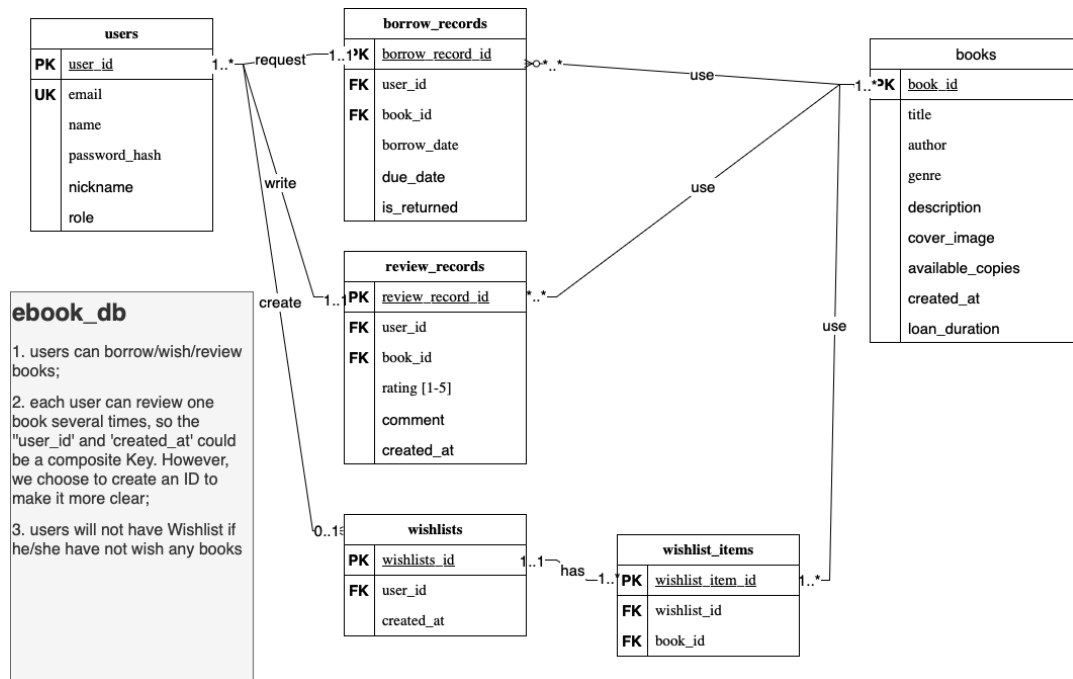






### 7.1.4 ER Diagram

A diagram showing the UML entity relationship diagram of the database design of eBook Loaning System. It shows how the user interacts with different modules and how these modules rely on external systems like databases, email services, and payment gateways.



## 7.2 Initial ideas for a GUI.

### 7.2.1: Version 1.0 of the Design

In Version 1.0 we initially used Figma to create the basic design. This looked at the basic functions, and how the user would interact with the eBook loaning tool.

#### Login and Registration Page: Figure 7.2.1.1 and Figure 7.2.1.2

##### Layout Details

- Users need a username and password to login. Users will also need an additional email to register.
- Two input boxes for the password box on the registration page to avoid mismatch
- Forgot Password will prompt user to input their email, which will have a new password sent to them
- Once the Login button is clicked it will redirect the user to the Home Page
- Login button will re-direct user to the register page in case they don't already have an account and vice versa

##### Design Highlights:

- Clean background with good colour scheme
- Simple and easy to follow

#### Homepage (Book Display): Figure 7.2.1.3

##### Layout Details:

- Top Navigation Bar: Fixed at the top of the page, includes a search bar, login/user info button (if the user is logged in), and book browse dropdown menu to view genres, dashboard button to see My Loans and Wish List

- Search Bar: Positioned in the center of the navigation bar page, with a keyword feature that shows related books.
- Book Display Area: Displays book cards, each card contains the book cover, title, and author
- Card Design: The book cover is centered, with the title, author, and description below.

#### Design Highlights:

- Ensure book covers are clearly visible, with appropriately sized images to avoid blurriness.
- Responsive design that adjusts the card layout to a single column on smaller screen sizes.

### Dashboard and User details: Figures 7.2.1.4; 7.2.1.5; 7.2.1.6

#### Layout Details:

- Navigation: Includes modules like 'My Books' and 'Wishlist'
- Borrowing Management: Displays the books the user has borrowed, with information such as book title, cover, borrowing date, and return date. Users can click to borrow or return books. Once the book has been returned it will be removed from the 'My Books' section
- Wishlist: Shows books in the user's Wishlist, and users can either add books to the borrowing list or write a review. (Same template as My Books in initial design)
- My Details: Users can change their email and username. Details are updated once the update button is pressed. Reset will clear their details
- Login & Security: Users can change their password. They will have to input current password and retype their new password twice

#### Design Highlights:

- Borrowing management should be simple and intuitive, with reminders when the borrowing period is nearing its end.
- Input fields are well-organized to avoid crowding.
- Keep the information concise and clear, highlighting the borrowed/liked book and related dates.
- Ensure the buttons are clearly visible to help users identify and complete actions.

### Admin Panel: Figures 7.2.1.7, 7.2.1.8, 7.2.1.9

#### Layout Details:

- Side Navigation Bar: Displays the 'Book Management', 'User Loans', 'Upload Books', 'Reviews Management' and logout button (takes the admin back to the login page).
- Book Management: Displays the book number, name, author, category, and status (available or loaned).
- User Loans: The admin can view the userID, currently loaned book number and loan status (available or loaned)
- Upload Books: Admin needs to fill out the author, title, book number, genre, and upload a cover image. Provide an "Save and Upload" button to add the book to the home page

#### Design Highlights:

- The admin interface should be clean, with clear functions and quick access to each module.
- When displaying data in tables, add sorting and searching functions to make it easier for the admin to find information.
- Provide pop-ups or confirmation dialogs to prevent accidental operations.

### Reviews Page: Figure 7.2.1.10

#### Layout Details:

- Pop up box that allows a user to write a review, add a rating and submit the review
- Reviews from other users included with rating
- Load more function to view more reviews

#### Design Highlights

- Reviews page should be clean with clear functions, and an option to click which star the user rates the book
- The page some seamlessly scroll down to read other reviews after load more function has been clicked

### 7.2.2: Version 2.0 of the Design

The initial design did not include a book details page, once this was completed, it was updated to correspond with the current pages already involved in implementation, so the design was slightly different.

#### Book Details Page: Figure 7.2.2.1

Layout Details:

- Book Cover: Positioned at the top of the page, displaying a large cover image, with the book title, author, category and description under it.
- Borrow Button: If the book is available for borrowing, show a "Borrow" button, which prompts success or failure messages upon clicking. (Not yet implemented during design layout but the functionality of this necessity was noted)
- Review Section: Displays a rating system (star rating), comment list

Design Highlights:

- Ensure the book cover image is clear, centered, and appropriately sized.
- Keep the book information layout simple to avoid clutter
- The review section should display ratings and comments clearly, with an easily visible review input box.

Figure 7.2.1.1

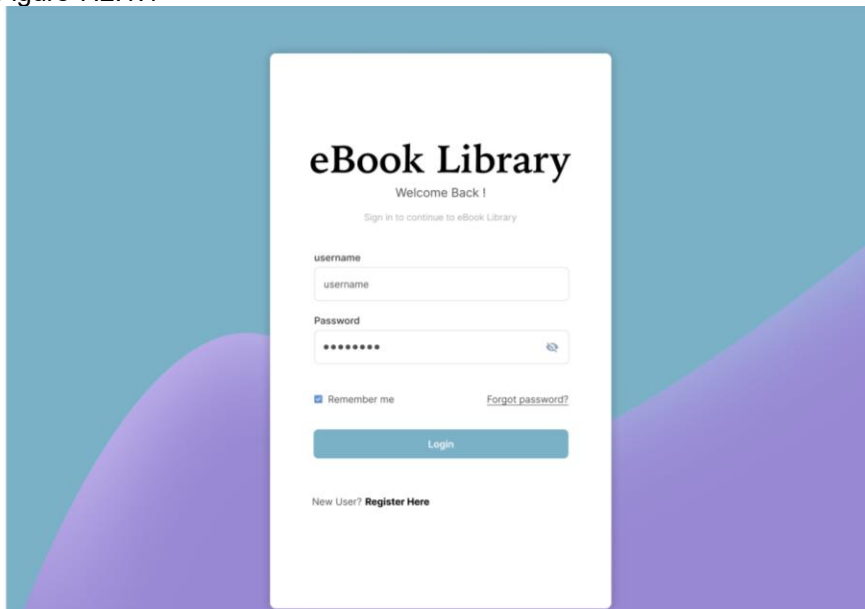
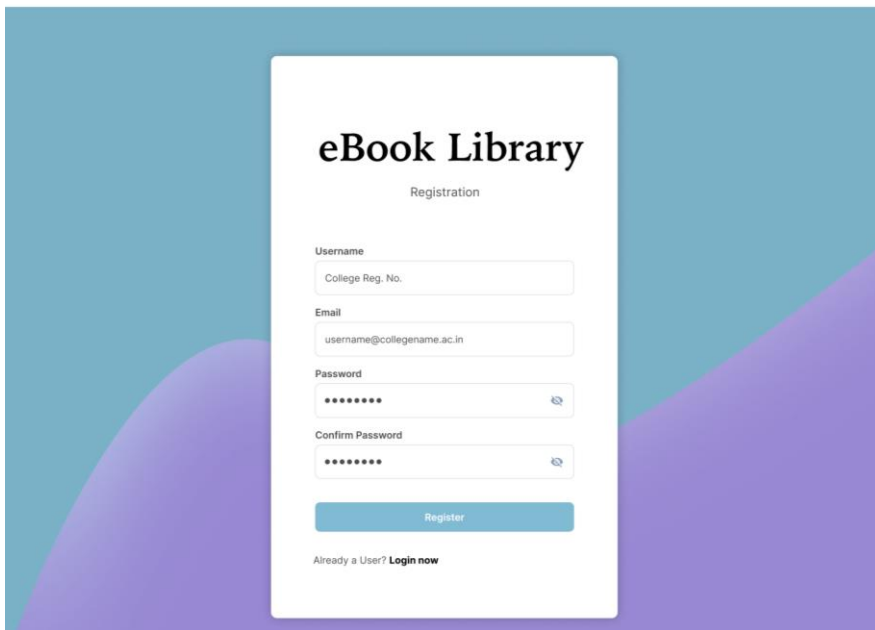


Figure 7.2.1.2



The registration form is centered on a white background with a blue and purple gradient. It includes fields for Username, Email, Password, and Confirm Password, each with a placeholder text. A 'Register' button is at the bottom, and a link to 'Login now' is below it.

**eBook Library**

Registration

Username  
College Reg. No.

Email  
username@collegename.ac.in

Password  
\*\*\*\*\*

Confirm Password  
\*\*\*\*\*

Register

Already a User? [Login now](#)

Figure 7.2.1.3

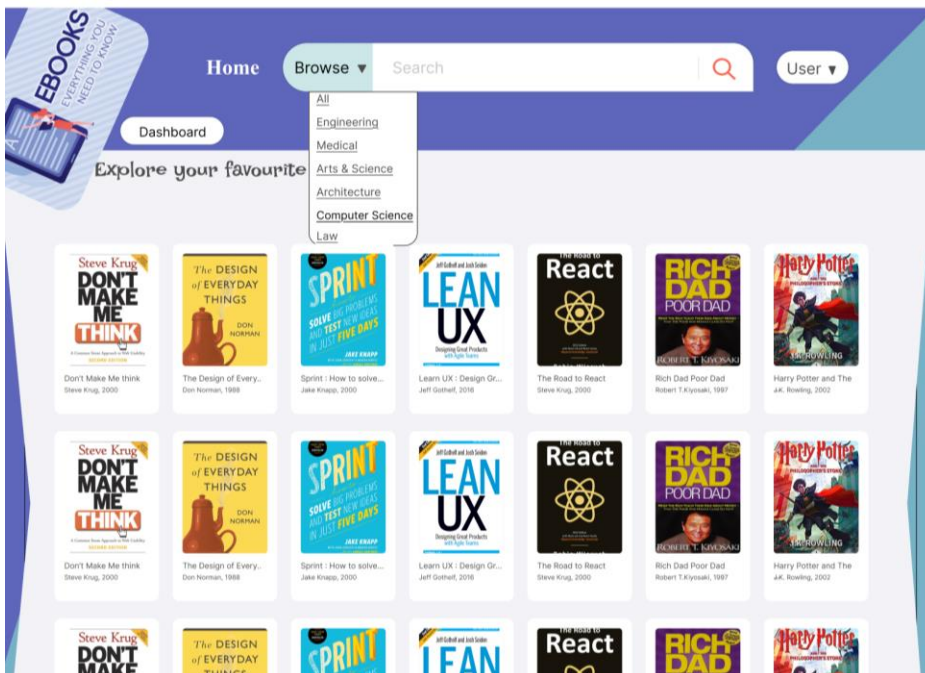
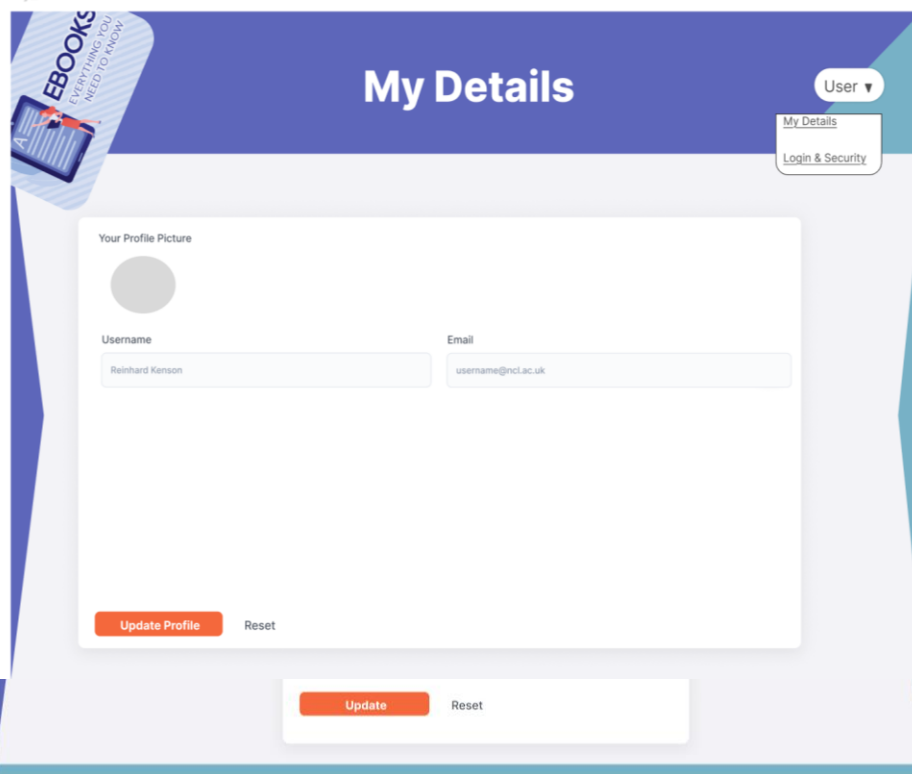


Figure 7.2.1.4



**My Details**

User ▾

My Details  
Login & Security

Your Profile Picture

Username: Reinhard Kanson

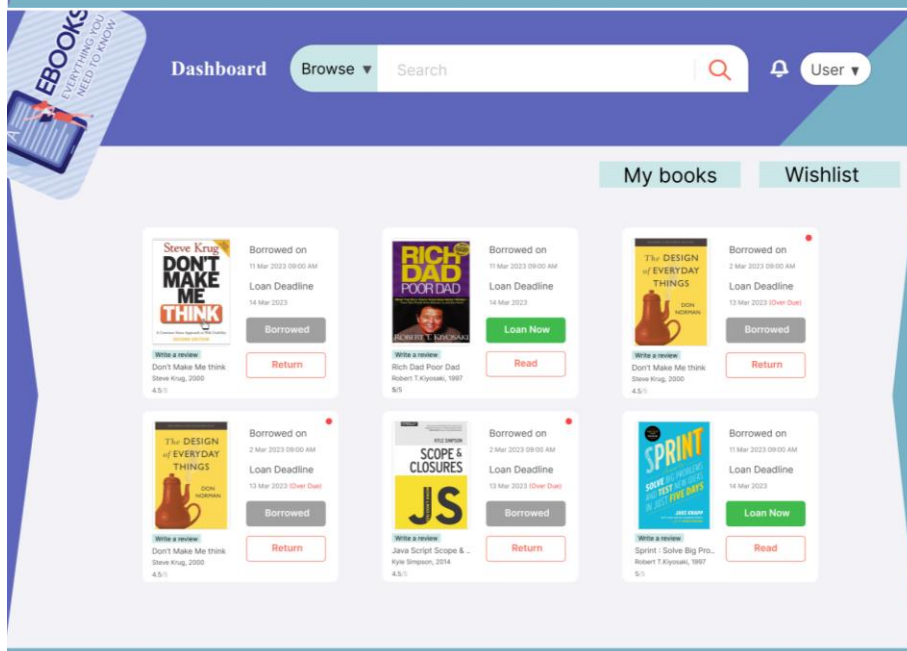
Email: username@nct.ac.uk

Update Profile Reset

Update Reset

Figure 7.2.1.5

Figure 7.2.1.6



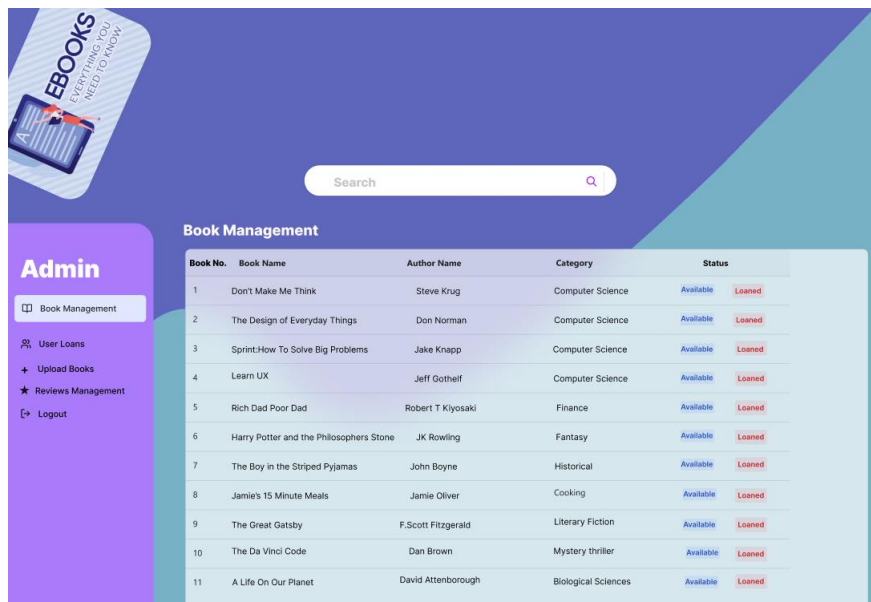
Dashboard Browse Search User ▾

My books Wishlist

Book Title	Borrowed on	Loan Deadline	Status	Action
Steve King: DON'T MAKE ME THINK	11 Mar 2023 09:00 AM	14 Mar 2023	Borrowed	Return
Rich Dad Poor Dad	11 Mar 2023 09:00 AM	14 Mar 2023	Loan Now	Read
The DESIGN of EVERYDAY THINGS	2 Mar 2023 09:00 AM	13 Mar 2023 (Over Due)	Borrowed	Return
The DESIGN of EVERYDAY THINGS	2 Mar 2023 09:00 AM	13 Mar 2023 (Over Due)	Borrowed	Return
SCOPE & CLOSURES JS	2 Mar 2023 09:00 AM	13 Mar 2023 (Over Due)	Borrowed	Return
SPRINT	11 Mar 2023 09:00 AM	14 Mar 2023	Loan Now	Read

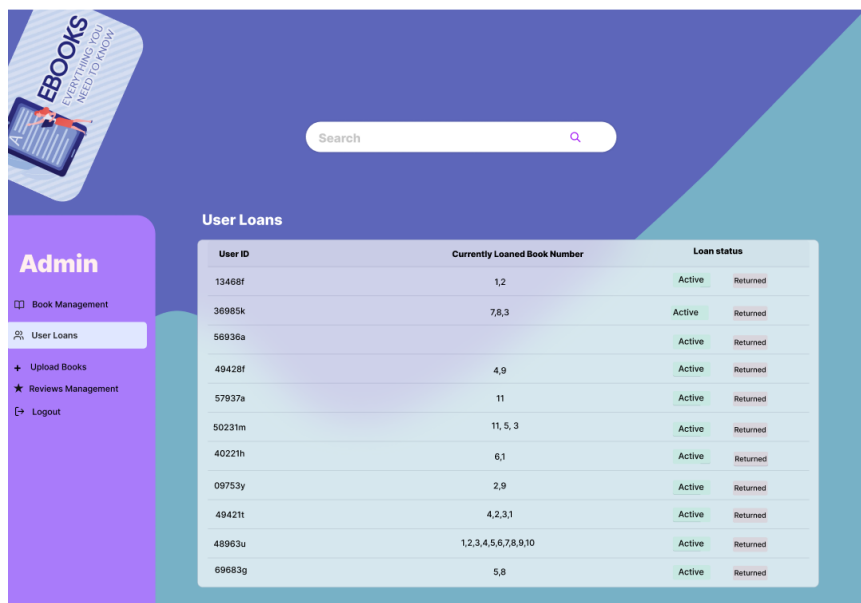
Figure 7.2.1.7





Book No.	Book Name	Author Name	Category	Status
1	Don't Make Me Think	Steve Krug	Computer Science	Available <span>Loaned</span>
2	The Design of Everyday Things	Don Norman	Computer Science	Available <span>Loaned</span>
3	Sprint:How To Solve Big Problems	Jake Knapp	Computer Science	Available <span>Loaned</span>
4	Learn UX	Jeff Gothelf	Computer Science	Available <span>Loaned</span>
5	Rich Dad Poor Dad	Robert T Kiyosaki	Finance	Available <span>Loaned</span>
6	Harry Potter and the Philosophers Stone	JK Rowling	Fantasy	Available <span>Loaned</span>
7	The Boy in the Striped Pyjamas	John Boyne	Historical	Available <span>Loaned</span>
8	Jamie's 15 Minute Meals	Jamie Oliver	Cooking	Available <span>Loaned</span>
9	The Great Gatsby	F.Scott Fitzgerald	Literary Fiction	Available <span>Loaned</span>
10	The Da Vinci Code	Dan Brown	Mystery thriller	Available <span>Loaned</span>
11	A Life On Our Planet	David Attenborough	Biological Sciences	Available <span>Loaned</span>

Figure 7.2.1.8



User ID	Currently Loaned Book Number	Loan status
13468f	1,2	Active <span>Returned</span>
36985k	7,8,3	Active <span>Returned</span>
56936a		Active <span>Returned</span>
49428f	4,9	Active <span>Returned</span>
57937a	11	Active <span>Returned</span>
50231m	11, 5, 3	Active <span>Returned</span>
40221h	6,1	Active <span>Returned</span>
09753y	2,9	Active <span>Returned</span>
49421t	4,2,3,1	Active <span>Returned</span>
48963u	1,2,3,4,5,6,7,8,9,10	Active <span>Returned</span>
69683g	5,8	Active <span>Returned</span>

Figure 7.2.1.9

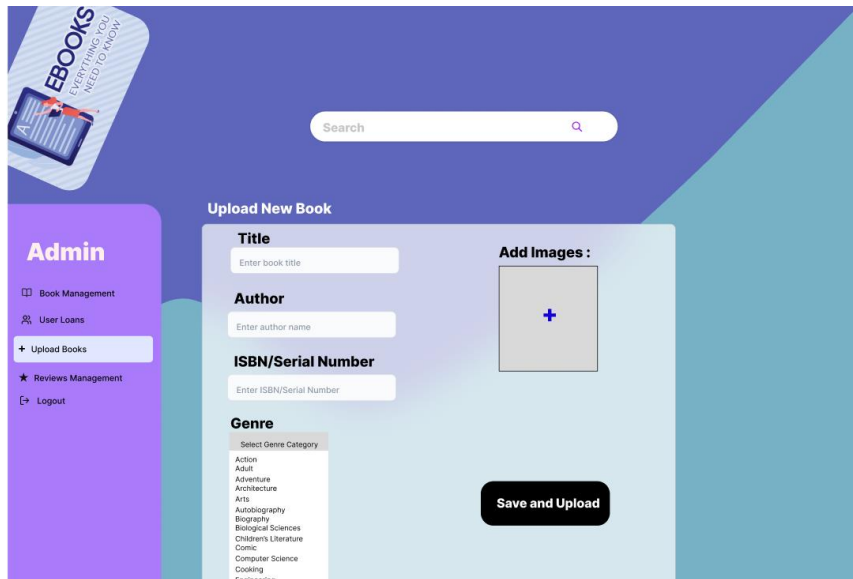
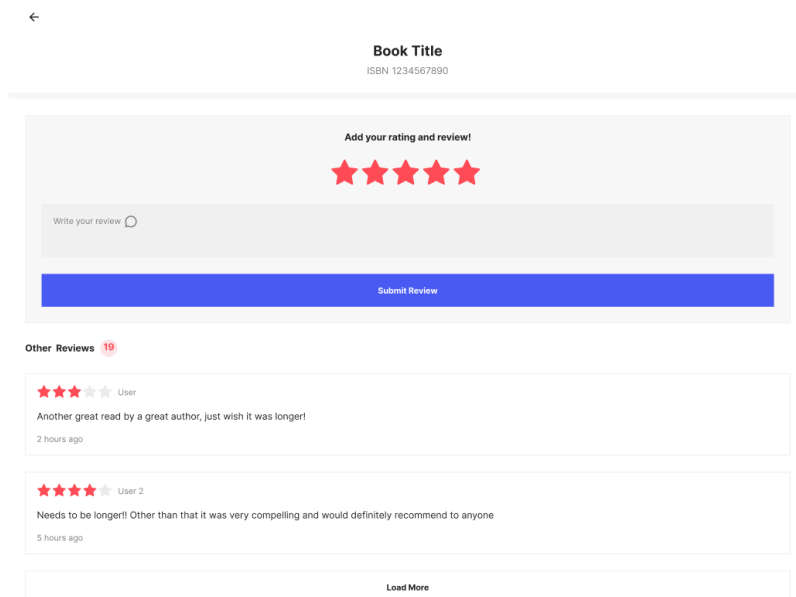
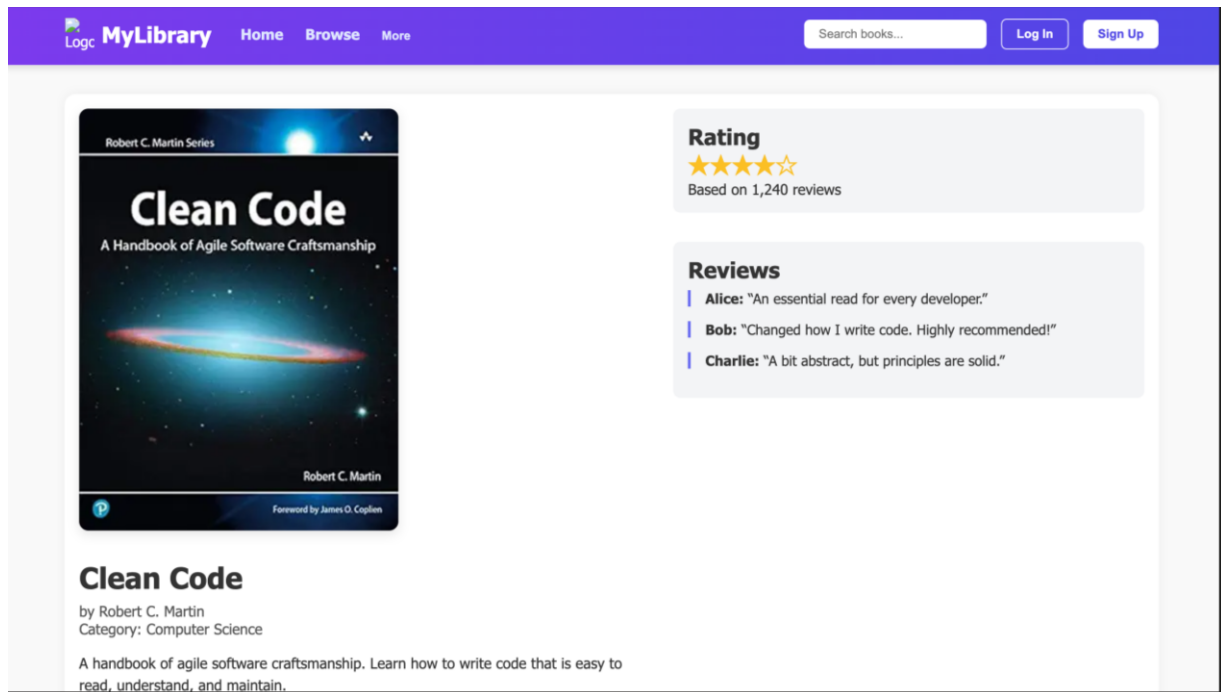


Figure 7.2.1.10



[Team 01]	Specification Document	[08/05/2025]
-----------	------------------------	--------------

Figure 7.2.2.1



### 7.3 Use cases

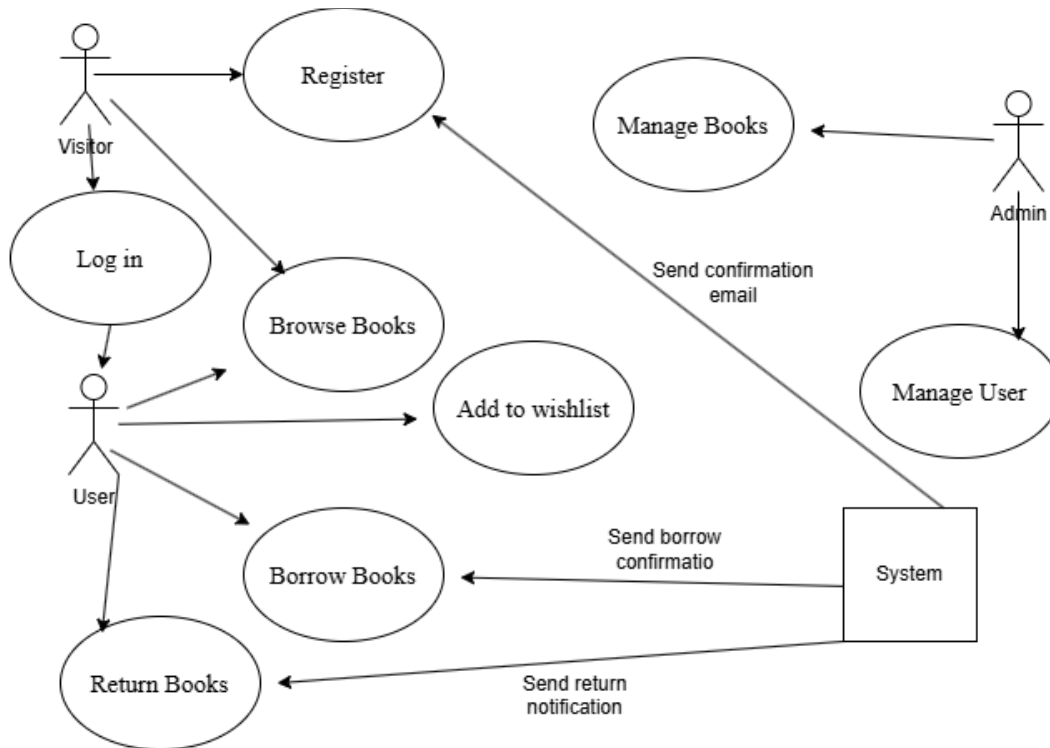


Figure 7.3.1 : Use Case Diagram of the eBook Loaning System

### Use Case Table – eBook Loaning System

Use Case ID	Use Case Name	Actor	Preconditions	Main Flow (Summary)	Alternate/Additional Notes
UC01	User Registration	Visitor	No existing account	Fill registration form → Submit → System creates account → Confirmation email sent	Invalid email/password → Show error
UC02	User Login	Registered User	Must have an account	Enter email and password → System validates → Redirect to dashboard	Wrong credentials → Show error
UC03	Browse eBook Catalogue	Any User	None	View book list with details (cover, title,	N/A

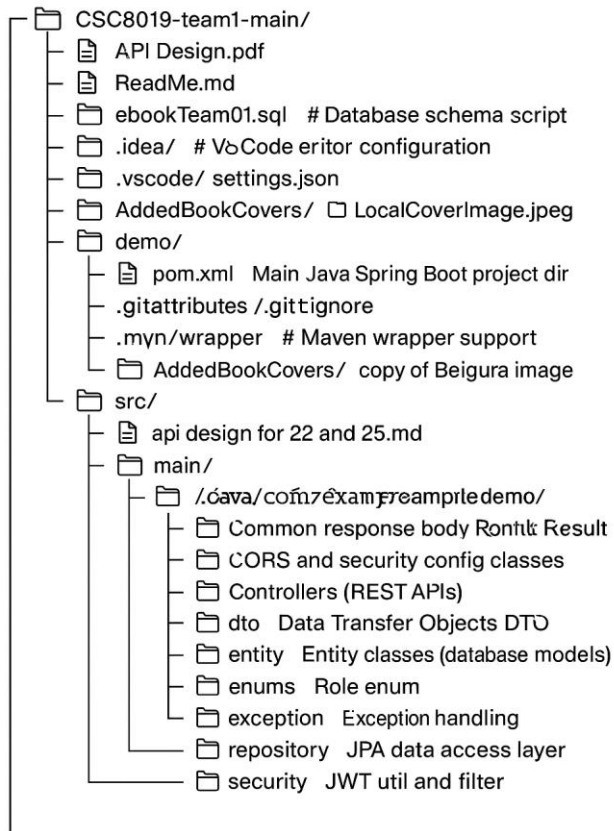
				author, category, etc.) → View individual book details	
UC04	Borrow an eBook	Registered User	Logged in, <10 active loans, book available	Click "Borrow" → System checks availability → Assigns book → Sends confirmation email	Book limit reached or no available copies → Show error
UC05	Cancel Loan	Registered User	Book is in active loans	View dashboard → Click "Cancel Loan" → System updates loan status → Sends cancellation email	N/A
UC06	Add to Wishlist	Registered User	Logged in	Click "Add to Wishlist" on book → System updates wishlist	N/A
UC07	Leave a Review	Registered User	Book must be borrowed before	Write rating/comment → Submit → System stores and displays review	Reviews visible to all users
UC08	Email Reminder for Loan Expiry	System	Loan nearing end (e.g., 2 days left)	System scans loan expiry dates → Sends reminder email	Runs as automated task daily
UC09	Admin Manage eBooks (Extension)	Admin	Admin logged in	Admin can add/edit/delete eBooks → System	Optional feature (Advanced/Admin Panel)

[Team 01]	<b>Specification Document</b>	[08/05/2025]
-----------	-------------------------------	--------------

				updates catalogue	
--	--	--	--	-------------------	--

## 8. Back end

### Project Architecture



### Admin Controller

```

71
72 */
73 @RestController
74 @RequestMapping("/api/admin")
75 public class AdminController {
76
77     @Autowired private AdminService adminService;
78     @Autowired private BookService bookService;
79
80     // download cover image to the local path
81     private static final String UPLOAD_DIR = "demo/AddedBookCovers"; // reletave path
82     // @Value("${file.upload-dir}")
83     // private String uploadDir;
84
85     // get all users
86     @GetMapping("/users")
87     @PreAuthorize("hasRole('ADMIN')")
88     public List<UserProfileResponse> getAllUsers() {
89         return adminService.getAllUsers().stream()
90             .map(UserProfileResponse::new)
91             .collect(Collectors.toList());
92     }
93
94     // delete user
95     @DeleteMapping("/users/{id}")
96     @PreAuthorize("hasRole('ADMIN')")
97     public ResponseEntity<String> deleteUser(@PathVariable Long id) {
98         try {
99             adminService.deleteUser(id);
100             return ResponseEntity.ok("user has been deleted successfully");
101         } catch (NoSuchElementException e) {
102             return ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());
103         }
104     }

```

#### Auth Controller

```

59
60 * - Designed by: Peilin Li
61
62 * - Reviewed by: Yunyi Jiang
63
64 * - Created on: 2025-04-27
65
66 * - Last modified: 2025-05-01 (Initial version)
67
68 */
69 @RestController
70 @RequestMapping("/api/auth")
71 public class AuthController {
72     @Autowired
73     private UserService userService;
74
75     @PostMapping("/register")
76     public ResponseEntity<String> register(@RequestBody RegisterRequest request) {
77         userService.register(request);
78         return ResponseEntity.ok("Registration successful");
79     }
80
81     @PostMapping("/login")
82     public ResponseEntity<> login(@RequestBody LoginRequest request) {
83         try {
84             AuthResponse response = userService.login(request);
85             return ResponseEntity.ok(response);
86         } catch (AuthenticationException e) {
87             return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("The username or password is incorrect");
88         } catch (RuntimeException e) {
89             return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
90         }
91     }
92
93
94     @GetMapping("/me")
95     public ResponseEntity<UserProfileResponse> getMe(@RequestHeader("Authorization") String authHeader) {

```

#### Book Controller

```

62 @RestController
63 @RequestMapping("/books")
64 public class BookController {
65     private final BookService bookService;
66     private final ReviewService reviewService;
67
68     public BookController(BookService bookService, ReviewService reviewService) {
69         this.bookService = bookService;
70         this.reviewService = reviewService;
71     }
72
73     @GetMapping
74     public List<Book> getAllBooks(
75         @RequestParam(defaultValue = "0") int page,
76         @RequestParam(defaultValue = "10") int size) {
77         return bookService.getAllBooks(page, size);
78     }
79     // getBookById
80     @GetMapping("/{bookId}")
81     public Book getBookById(@PathVariable int bookId) {
82
83         return bookService.getBookById(bookId);
84     }
85
86     @GetMapping("/{id}/details")
87     public BookDetailResponse getBookDetails(@PathVariable int id) {
88         Book book = bookService.getBookById(id);
89         List<ReviewRecord> reviews = reviewService.getReviewsByBook(id);
90         return new BookDetailResponse(book, reviews);
91     }
92
93     // add api: getAllGenres
94     @GetMapping("/genres")
95     public List<String> getAllGenres() {
96         return bookService.findDistinctGenres();
97     }
98

```

### Borrow Service

```

4 import java.util.List;
5
6 /**
7  * Service interface for managing borrowing operations.
8  * Handles borrowing, returning, and retrieving borrowing history of books.
9  *
10  * Author: Guanyuan Wang
11  */
12 public interface BorrowService {
13
14     /**
15      * Borrows a book for the given user.
16      *
17      * @param userId the ID of the user borrowing the book
18      * @param bookId the ID of the book to be borrowed
19      */
20     void borrowBook(int userId, int bookId);
21
22     /**
23      * Returns a borrowed book for the given user.
24      *
25      * @param userId the ID of the user returning the book
26      * @param bookId the ID of the book to be returned
27      */
28     void returnBook(int userId, int bookId);
29
30     /**
31      * Retrieves the borrowing history for a given user.
32      *
33      * @param userId the ID of the user
34      * @return list of borrow records
35      */
36     List<BorrowRecord> getBorrowHistory(int userId);
37
38     /**
39      * Retrieves all borrow records in the system.
40      */

```

### Email Service



```

48
49
50 @Service
51 public class EmailService {
52
53     @Autowired
54     private JavaMailSender emailSender;
55
56     // send email
57     public void sendSimpleMessage(String to, String subject, String text) {
58         SimpleMailMessage message = new SimpleMailMessage();
59         message.setTo(to);
60         message.setSubject(subject);
61         message.setText(text);
62         message.setFrom("jiangyunyi000@gmail.com");
63         emailSender.send(message);
64     }
65
66
67     public void sendRegistrationConfirmation(String userEmail) {
68         String subject = "Registration Confirmation of Teams01's EBook library";
69         String text = "Thank you for registering Team01's EBook System! Your account has been created successfully";
70         sendSimpleMessage(userEmail, subject, text);
71     }
72
73     public void sendBorrowConfirmation(String userEmail, String bookTitle) {
74         String subject = "Borrow Confirmation";
75         String text = "You have successfully borrowed the book: " + bookTitle + ".";
76         sendSimpleMessage(userEmail, subject, text);
77     }
78
79     public void sendReturnConfirmation(String userEmail, String bookTitle) {
80         String subject = "Return Confirmation";
81         String text = "You have successfully returned the book: " + bookTitle + ". Thank you!";
82         sendSimpleMessage(userEmail, subject, text);
83     }
84 }
85

```

## User Service

```

* - Last modified: 2025-05-03 (Initial version)
*/
@Service
public class UserService {

    @Autowired private UserRepository userRepository;
    @Autowired private PasswordEncoder encoder;
    @Autowired private JwtUtil jwtUtil;
    @Autowired private AuthenticationManager authManager;
    @Autowired private EmailService emailService;

    // user send email when register

    // register
    public void register(RegisterRequest request) {
        if (userRepository.existsByUsername(request.getUsername())) {
            throw new RuntimeException(message:"use has been registered");
        }
        User user = new User();
        user.setUsername(request.getUsername());
        user.setPassword(encoder.encode(request.getPassword()));
        user.setEmail(request.getEmail());
        user.setRole(Role.USER); // default
        User savedUser = userRepository.save(user);
        // send email when register successfully
        emailService.sendRegistrationConfirmation(savedUser.getEmail());
    }

    public Integer getUserIdFromToken(String token) {
        return getCurrentUser(token).getId().intValue();
    }

    public AuthResponse login(LoginRequest request) {
        authManager.authenticate(new UsernamePasswordAuthenticationToken(

```