Project Report

Development of Web-based Library Management System

SUBMITTED BY

Rohan Sinha

Roll No: ASH2011050M

Session: 2019-2020

Year: 04; Term: 02

Department of Information and Communication Engineering

PROJECT SUPERVISOR

Dr. Md. Ashikur Rahman Khan

Chairman & Professor

Department of Information and Communication Engineering

Noakhali Science and Technology University

Noakhali-3814, Bangladesh

July 2025

DECLARATION

This project report is submitted to the Department of Information and Communication Engineering, Noakhali Science and Technology University, in partial fulfillment of the requirements for having the B.Sc Engineering degree in ICE. This is also needed to certify that the project work is under the B.Sc in Engineering Course of department of ICE, NSTU titled "ICE-4218: Project and Thesis". So, I declare that this project report has not been submitted elsewhere for the requirement of any kind of degree, diploma and publication.

Rohan Sinha

Student of Bachelor of Science,

Department of Information and Communication Engineering,

Noakhali Science and Technology University,

Sonapur – 3814, Noakhali, Bangladesh.

ID : ASH2011050M

Session : 2019-20

ACCEPTANCE

The project report is submitted to the Department of Information and Communication Engineering, Noakhali Science and Technology University, Noakhali in partial fulfillment of the requirements for having the B.Sc Engineering degree in ICE.

Dr. Ashikur Rahman Khan,

Professor,
Department of Information and Communication Engineering

Noakhali Science and Technology University.

Sonapur – 3814, Noakhali, Bangladesh.

ABSTRACT

Library management system is a project which aims in developing a computerized system to maintain all the daily work of the library .This project has many features which are generally not available in normal library management systems like facility of user login and a facility of teachers login. It also has a facility of admin login through which the admin can monitor the whole system .It also has facility of an online notice board where teachers can student can put up information about workshops or seminars being held in our colleges or nearby colleges and librarian after proper verification from the concerned institution organizing the seminar canada it to the notice board . It has also a facility where students after logging in their accounts can see the list of books issued and its issue date and return date and also the students can request the librarian to add new books by filling the book request form. The librarian after logging into his account i.e. admin account can generate various reports such as student report , issue report, teacher report and book reportOverall this project of ours is being developed to help the students as well as staff of library to maintain the library in the best way possible and also reduce the human efforts. The whole system will be developed by Iterative Model. The complete project is developed using HTML (Hyper Text Markup Language), CSS (Cascading Style Sheet), React.js, Java, Spring boot, PostgreSQL, REST API's.

Keywords: Iterative model, Computerized System, Authentication, HTML, CSS, React, Java, Spring boot, PostgresQL, REST API's.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

A library is a collection of books that its members and those of affiliated institutions can use, along with perhaps other resources and media. Libraries can be actual places, virtual spaces, or both that offer digital (soft copies) or physical (hard copies) resources. A library's collection often consists of printed items that are available for loan as well as a reference section of publications that are only available for use on library property. Commercial releases of movies, TV shows, other video records, radio, music, and audio recordings are examples of resources that can be found in a variety of formats. These consist of CDs, DVDs, Blu-rays, cassettes, and other relevant media like microform. They might also make information, music, and other materials stored on bibliographic databases accessible.[1]

The earliest attempts to arrange document collections marked the beginning of libraries' history. Archives of the first writing system, the cuneiform-scripted clay tablets found in Sumer, some of which date to 2600 BC, made up the first library. In the fifth century BC, private or individual libraries composed of written books first developed in classical Greece. The largest libraries of the Mediterranean world were still those of Constantinople and Alexandria in the sixth century, toward the end of the Classical era. [1]

Within their territories, the Fatimids (r. 909–1171) also had numerous excellent libraries. According to the historian Ibn Abi Tayyi, their palace library was a "wonder of the world" and likely had the greatest collection of books in existence at the time. Along with brutal atrocities, the burning of libraries has always been a crucial part of conquerors' plans to erase any traces of the defeated community's written past. The Mongol killing of the Nizaris at Alamut in 1256 and the burning of their library, "the fame of which" the conqueror Juwayni claims "had spread throughout the world," are notable examples of this. [1]

A significant change occurred in the library system at the beginning of the twentieth century. The books were made available to everyone, even those living in rural areas. Every task in a traditional library is carried out and overseen by hand. The maintenance of traditional accounts is inefficient and time-consuming due to the vast amount of resources and library storage. As technology develops, libraries need to adapt to new developments and use technology to efficiently manage their holdings and operations. By employing computers to manage daily operations, technology can help transform the traditional library

into a digital one.

1.2 Background Information

An automated system called the Online Library Management System oversees the different operations of the library. It manages every procedure at the library. Everything is automated, including check-in and checkout, material borrowing, item updates, and resource and book organization. Any educational institution or organization, whether public or private, can use the Online Library Automation Software. [2]

Thus, there are many benefits to employing an online library management system, including:

I. Automates library operations
II. robust search interface
III. keeps up with library records and boosts productivity
IV. Software that is both scalable and secure
V. Simple access to all library materials at any time and from any location
VI. Software that is affordable
VII. Software that is simple to use
VIII. Tracking library resources is simple.
IX. An interactive dashboard to improve user experience
X. Capabilities for collaboration
XI. It lowers the data errors.

1.3 Problem Statement

All of the information details in the seminar library of the department of information and communication engineering are manually maintained and stored in register books. Therefore, managing the entire library system by hand has some significant disadvantages. Here are some important points to note:

I. It's possible that the register book or any other crucial data would be lost.
II. It is possible to tear the book into many pieces.
III. It takes a lot of time to manually find out any student information.
IV. No search function is available.
V. It takes up time and space.
VI. The manual registration approach is very labor-intensive.
VII. Since the summary report is not method-based, it may contain inaccuracies.
VIII. Evaluating the fine for being past due is very difficult.

1.4 Motivation

It takes a lot of time and effort to keep up with all the procedures when we try to look for books in the Seminar library. In addition, we must look through every book to find a specific book.

Both the librarian and the patrons will benefit greatly from this library management system project. For example, rather than storing the information in a register book, the librarian can store it in an electronic system. As a result, there is no chance of data loss, computerized methods are more efficient for handling book issues and returns, and the librarian can send out notifications to the user when a book is returned and users can look up books, their problems, the books' return date, and other information. Therefore, the project will enable the librarian and the user to run a more efficient library system.

1.5 Objectives

The objectives of this project are to design and develop a computer-based seminar library that can manage the daily activities of the library in more effective and efficient ways. The others objectives of this project are listed below:

I. To maintain the details of books and library members
II. To provide security of the data in the system.
III. To enhance the productivity of librarians, library staff and users.
IV. To reduce the management time and the managing cost.
V. To provide mailing services in the system.
VI. To provide access to members to their recent activities

1.6 Scope

The document only covers the requirements specifications for the Seminar Library Management system. Such as,

I. The other component of the Seminar library management system is not mentioned in this publication.
II. This article also includes links to all external interfaces and dependencies.
III. Information including book details, new book additions, lost book deletions, and fines for holding onto a book for more than 15 days beyond its issuance

date are all provided by the system.

IV. The addition of new books to the library can be requested by patrons.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Understanding the fundamentals of computer-based library management systems requires reading and studying published material written by experts in the field. The goals of this review are to analyze, examine, and gain experience with the documents' production and archival processing. Based on a careful analysis of the literature on computerized library management systems, the review provides an outline of the main concepts associated with the creation of an electronic records management system from the viewpoint of published specialist.

2.2 Related Work

Balasubramani et al. (2010) updated the book and resource database of the Chemistry Library at Bharathidasan University using KOHA Open-source software. They managed flow operations like book borrowing and return using open-source software coupled with the KOHA library. Their objective was to improve customer service by modernizing library operations. For this project, they selected KOHA, a complete software program appropriate for both small and large businesses. They used KOHA Open-Source Software to successfully develop the automated library management system in spite of challenges. His research was limited by the following:

I. Inadequate Infrastructure Resources
II. Limited financial resources and little environmental support. [7]

For the State University of Nigeria, Maureen et al. (2011) developed an automated library management system. Using SQL Enterprise Manager, they constructed the database for the system, which contained tables for Catalogue, Circulation, Serial, User Account, and Library User. Activities including software development, testing, debugging, hardware configuration, and staff training were all part of the system implementation process. The system underwent extensive testing, including Alpha and Beta testing phases, and a two-tier software architecture was employed [9].

Bhupendra et al. (2011) created an online library administration system to automate library activities. This system provides tools to help users with tasks including stock management, report generation, and transaction entry. The authors carried out a feasibility assessment of the current system in order to identify issues and obtain a better grasp of how the library

operates. The system was composed of three parts: inventory management, transaction tracking, and report generation. [10]

In order to efficiently manage a variety of library duties, Shivkumar et al. (2012) described a library management system (LMS) as multitasking, multi-user integrated software that can operate on a single computer or in a client-server multi-platform configuration. Given the size of a library's book collection and patron base, manual management is not feasible. The study's library management system has a number of features and modules, such as:

I. Circulation and Acquisition
II. Cataloging OPAC [Online Public Access Catalog]
III. Verification of Serials Administration Stock

Tripathi et al. (2012) developed a Java-based library management system that prioritizes the library's core functions, including book checkout and return, member and book administration, and information updating. Users can add, search for, update, issue, and return books with ease thanks to their program, which takes care of these duties. It features a search capability, larger storage, quicker database access, fewer mistakes, and an intuitive user interface. However, it lacks features like scanning capability and a notification system [5].

A library management system was created by Kumar et al. (2014) to digitally track day-to-day library operations. User and instructor logins, together with an administrator login for supervision and management, are some of their system's unique features. Their design was meant to achieve certain objectives. The following objectives guided the system's design:

I. Improved control and performance.
II. Cut costs.
III. Make the most of your time.
IV. An online notice board substitute.

Liu et al. (2018) conducted study on a C library management system that was functioning on the DOS operating system. Administrators, students, and unregistered visitors were the three categories of users that could access the system. It allows you to add, remove, amend, and query library resources. Users could check out and return books by logging in, but administrators could manage the books and quantity. Visitors might browse the book database but not check out any books. However, there were problems with the system, such as insufficient data storage and the inability to support more users and books. It was inconvenient for users to have to authenticate themselves again when borrowing books, and data updates were not saved for a long time [2].

A.P. et al. (2020) used dot technology to convert a manual library system to a digital one.The front-end program was developed in C#, but the database was maintained in SQL. They discovered issues with the data and poor performance in the outdated library system. With complete control over their system, the administrator could add books, members, issue and return books, and change data. Books may be checked out by those who have their login information. However, their system failed to notify users when the return window for books had elapsed [3].

To replace the manual library administration system at Asmara Community College of Education (ACCE), Araya et al. (2020) created a web-based system. To eliminate paperwork, the system was designed and developed using HTML, PHP, Java, and MySQL. Managing book additions, removing duplicate registrations, adding new members, and user login verification were among the subjects discussed. Despite serving both patrons and workers, the library had several drawbacks, such as the need to add books that patrons requested [4].

All of the aforementioned works are significant. With its various functions, an automated library system can replace a human one. Additionally, there are numerous mistakes or gaps in the work. I create and assess a model for an online library management system as part of this study. I believe that this study can close the gaps mentioned in the section that was previously stated. The goal of this project is to develop an automated system that will help university libraries maintain accurate transaction records while safeguarding user data. Additionally, to facilitate the search by displaying the precise location of library resources and maintaining a record of all checked-out and returned book.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The methodology provides a well-articulated plan for the development of the online automated library management system, where the whole focus will be of teaching-friendly interfaces, devised to look after the whole management of library operations, and greater accessibility for the user. The system shall address, primarily, three key user roles: administrators, teachers, and students, while providing such key operations as book transaction, faculty services and maintenance, and reports.

The system shall address, primarily, three key user roles: Administrators, Teachers and Students.

3.1.1 Admin Module

The administration module provides a variety of administrative controls throughout the library functions.

Features:

I. **Admin login:** An easy and reliable way to log into the system, with his unique user ID and password.
II. **Validation of users:** Admins can verify and approve student and teacher registrations.
III. **Reporting Management**: To generate reports.
IV. **Student Data:** Information on transactions that a student has availed during his whole stay.
V. **Teacher Data:** Information on how many books a teacher has issued and what departmental contribution a specific teacher has made.
VI. **Book Inventory**: Details on how many of the library's books are still not borrowed or in stock.
VII. **Transaction Data:** Depending on their transactions, the list includes: issued, returned, and overdue books.

VIII. **Objective:** To allow the library system admin to operate more effectively and maximize its operations within the library.

3.1.2 Teacher Module

This module facilitates teachers to access library resources and, through the platform, promotes departmental contributions.

Features:

I. **Teacher login**: By the means of a unique login ID and password, the system can allow only an authenticated teacher to log into the system.
II. **Password Recovery**: A feature developed for the teacher to recover his or her password with no chance of the administration front's breach.
III. **Email**: Email for contact with teachers.
IV. **Isbn(International Standard Book Number):** This requires the entry of identifying the book and loaning it to the teacher.
V. **Event Addition**: This encourages teachers to log academic events under their department's contribution.
VI. **Objective**: At the same time with ease of access to library resources, allow for improved cooperation within the department.

3.1.3 Student Module

This module facilitates teachers to access library resources and, through the platform, promotes departmental contributions.

I. **Student login**: By the means of a unique login ID and password, the system can allow only an authenticated teacher to log into the system.
II. **Password Recovery**: A feature developed for the teacher to recover his or her password with no chance of the administration front's breach.
III. I**sbn(International Standard Book Number)**: This requires the entry of identifying the book and loaning it to the student.
IV. **Session :** It represents the academic session the student is currently enrolled in.
V. **Image**: A profile picture for each individual student.
VI. **Contacts** : Email for contact with students and a contact number.

VII. **Objective**: At the same time with ease of access to library resources, allow for improved cooperation within the department.

It will provide such key operations as book transaction, book maintenance , report generation.

3.1.4 Transaction Module

In this section the date of issue book and return date will be shown and admin can send warning notification to the students.

I. **Issue book** : It used to borrow books from the library. It includes search for a book, verifying eligibility and updating inventory.
II. **Return book** : It includes returning the borrowed book and fine calculation if needed.
III. **Availability:** It includes availability of a book and notify about the book duration.

3.1.5 Book Maintenance Module

Here the key operations are performed as,

I. **Add new book** : Adding new book to the library.
II. **Delete book** : Deleting a specific book.
III. **Search book** : Searching for a required book.
IV. **Update books information** : Updating book information according to the transaction.

In this, the admin can add a new book, delete or update the information of the existing book.

3.1.6 Report Generation Module

It will provide statistical and other important information about the transactions, availability etc.

I. **Student Report** : Provides a number of students enrolled for a specific book and issue report.
II. **Teacher Report** : Provides number of teachers enrolled for a specific book and issue report.
III. **Book Information Repor**t : Provides book details and other operations.
IV. **Book Transaction Report :** Provides book transaction report.

3.2 Requirement Specification

A requirement specification gives a concise explanation of the objectives that should be pursued by a system, product, or project while accounting for both functional and non-functional requirements.

3.2.1 Functional Requirement

The system encompasses a number of core services that facilitate the functions and activities of a library and make user's work easier. The process of registration, or sign up, is done in a manner that considers varying users and use contexts. Librarians are required to fill in their name, phone number and email id, while students and teachers have to fill in the same details but with a password that is meant to be secure. The librarian, upon entering his or her details during the registration process, goes through an authentication check where the information provided is validated. When the authentication process is completed successfully, the system confirms the user's state of registration and generates a membership number which it forwards through the user's provided email address. If at any point mistakes are found, an error message is generated to assist the user.

At login, users have to provide their email and password. Subsequent to submitting the correct password, a user has been authenticated in the system and is able to use the features of the system. Among these features is the ability for users to see their currently issued books and the dates when those books are due back. The user can find books by typing the name or associated keyword and the system will generate a list of all matching books. When the user decides to select a book to issue, the system verifies whether the book is available or not. If the book is available for the set period, it is issued and the user gets a confirmation.The system will issue an error message if the user attempts a transaction that cannot be fulfilled. In the same manner, a user can enter the book title and renew books that are about to be due and the system will confirm the renewal.

To return books, the user is to take the book to the library and the system will automatically change the list of rented books. For returned books that are overdue, the system applies a penalty of BDT 10 for each day past the due date and gives the user all details regarding penalties imposed. In addition, the system enables librarians to perform book inventory management functions of addition and deletion. While adding books, they capture the book name, author, edition and quantity, and the system gives them confirmation once it is complete. For the deletion of books, the same data with the exception of quantity is needed and the system processes the changes to the inventory.

Indeed, the technology application supports the effective management of library activities and enables appropriate and effective communication between users, be they student teachers, or librarians.

3.2 2 Non-Functional Requirements

Non-functional requirements define the important aspects and constraints as well as the environmental conditions a system has to comply with besides normal operation. The system interface should be designed with aesthetics which facilitate ease of use with seamless navigation for users and administrators alike and easy login and input provision. With regards to system availability, the system shall function without interruption, 24 hours a day, 7 days a week throughout the year for 100% operating time. The system should also employ mechanisms that allow speedy recovery from any failure that may happen, thus facilitating operational efficiency. The system is expected to provide reliable real time data without interruption for multi collaborative tasks which makes accuracy very important. The system must also perform at a high level by automatically updating data and responding to user requests within mere seconds, candidates should expect all on screen information to load within five seconds. Equally as important, error handling is where the system manages both anticipated problems and the unforeseen while enabling smooth operation and data integrity. Most importantly, security has to be tight and only authorized persons such as the librarian are allowed to make changes to the databases. The system must also support a vast number of books and us.

3.2.3 Process Model

A process model is a structured framework that describes the steps, activities, and workflows involved in achieving a specific objective, typically in the context of software development, business operations, or project management. It provides a systematic approach to planning, executing, monitoring, and improving processes to ensure efficiency, consistency, and quality.

A process model is essentially a carefully structured framework that outlines in detail the steps, activities, and rest of the workflows needed to accomplish a single objective, such as software development, business operations or project management. Aid in the implementation of appropriate mechanisms for planning, implementation, evaluation, and improvement of processes for the achievement of the desired level of efficiency, uniformity and quality is provided.

There are various process models, such as -

Waterfall Model
A set of requirements is captured and defined, then proceeds to system design, followed by system implementation, testing, deployment and finally, maintenance. Each phase must be completed before proceeding to the next. Best suited for project work where the

requirements are clearly understood.

Iterative Model
The resulting system is built and enhanced through the expansion of more and more refined prototypes, until the final product is produced.

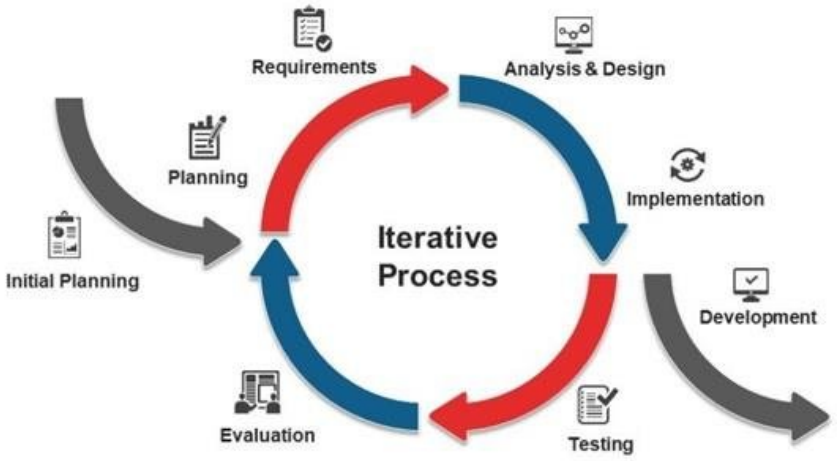

Figure 3.2 : Iterative Model Process

Incremental Model
The project is divided into several modules that are small enough in size to be developed, delivered and used independently. Each module is an incremental improvement to the existing system.

Agile Model
An approach that defines ways of working to enhance flexibility and iteration by applying collaboration to drive customer satisfaction. Includes Scrum, Kanban, and Extreme Programming (XP).

Scrum
Focuses on specific iterative development processes within Agile, where defined roles, sprints, and daily stand-ups are practiced (Scrum Master, Development Team, Product Owner).

Reason of using Iterative Model

The Iterative Model is particularly useful for projects with multiple versions, rapid delivery, and systematic changes. The development of the system occurs in a series of smaller cycles during which testing and stakeholder feedback is necessary. This provides early detection of issues which improves quality and covers the changing requirement. The model also decreases risks, costs due to rework, and enables early deployment of core functionalities. It's best-suited when there are unknowns in requirements, and those requirements are subject to changes, as well as when the user has to be involved. The model achieves a greater

adaptability, scalability, and a user-focused outcome by refining the product iteratively.

3.4 System Design

The requirements and problem analyses determine the design of the system. Based on this, I will attempt to create this system so that users may execute it more quickly.

Three components make up the system design:

I. Design of use case diagrams
II. Diagram E-R
III. Diagram of the activity flow

Use-Case Diagram

The purpose of a use case diagram is to define the interactions of the users and the system outlining the goals that need to be achieved. All of these are important in capturing what the system should offer at a high level. It enables the stakeholders, developers, and designers to understand the system better by clarifying the functionality and the overall scope through identification of the key actors, use cases and their dependencies. Also, user interactions, and system limits are made clear, which leads to effective communications, alleviates complex processes, and makes sure none of the user needs are forgotten in the development process.

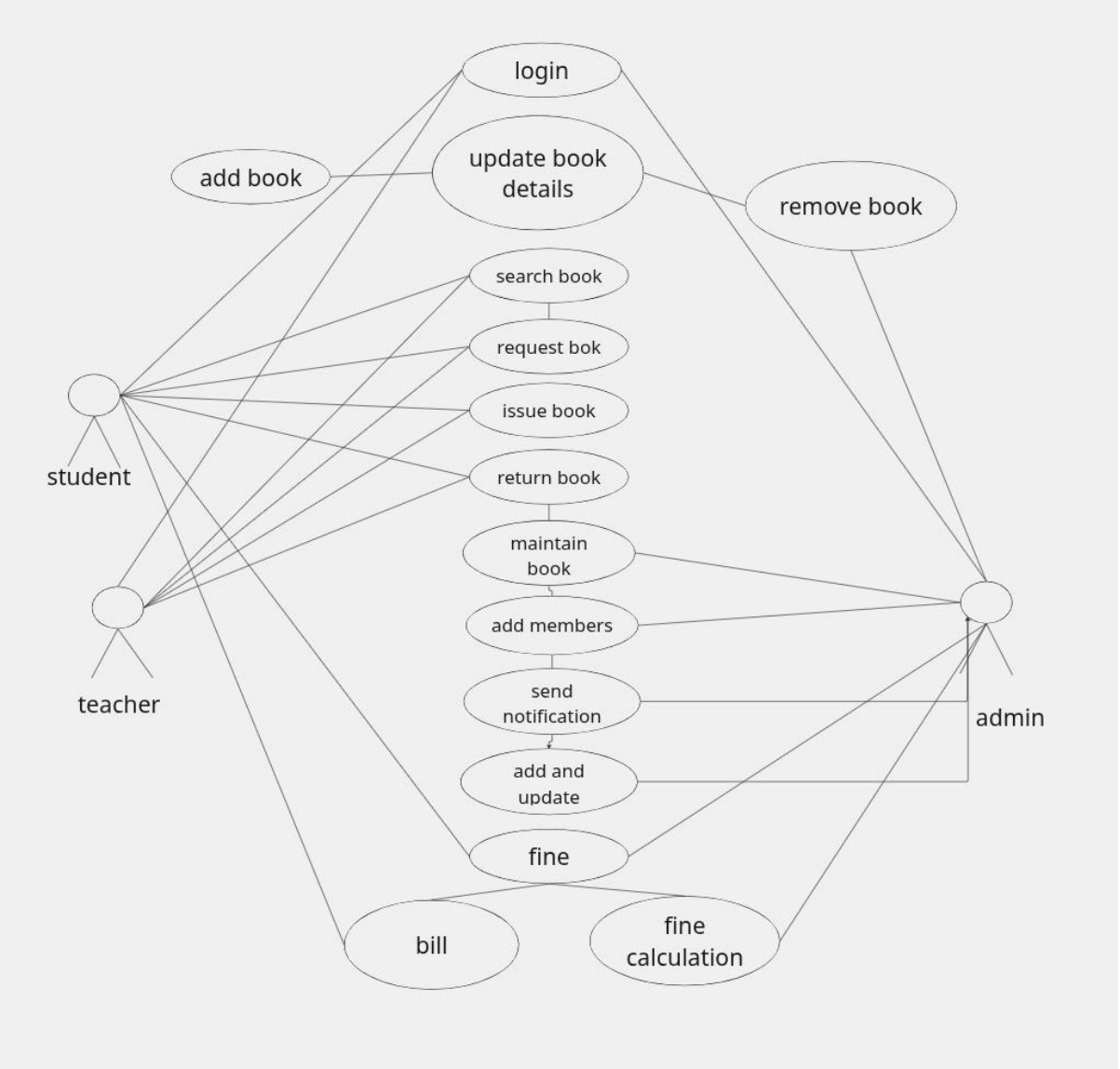

Figure 3.3: Use case diagram for Seminar Library Management system

E-R Diagram

An entity relationship or E-R diagram is one of the primary components of designing a database, which is prepared to indicate what a database comprises and how the pieces of data relate to each other. It spells out with clarity the different entities (objects or concepts), attributes (properties of entities) and relations (associations of different entities). E-R diagrams are beneficial to database designers as well as developers because they provide more description with regards to the data and the constraints required making properly organized and well-defined database schema. They are also helpful in establishing proper normalization and data integrity by specifying primary keys, foreign keys, and cardinality, thus making them extremely important for effective and efficient database systems development.

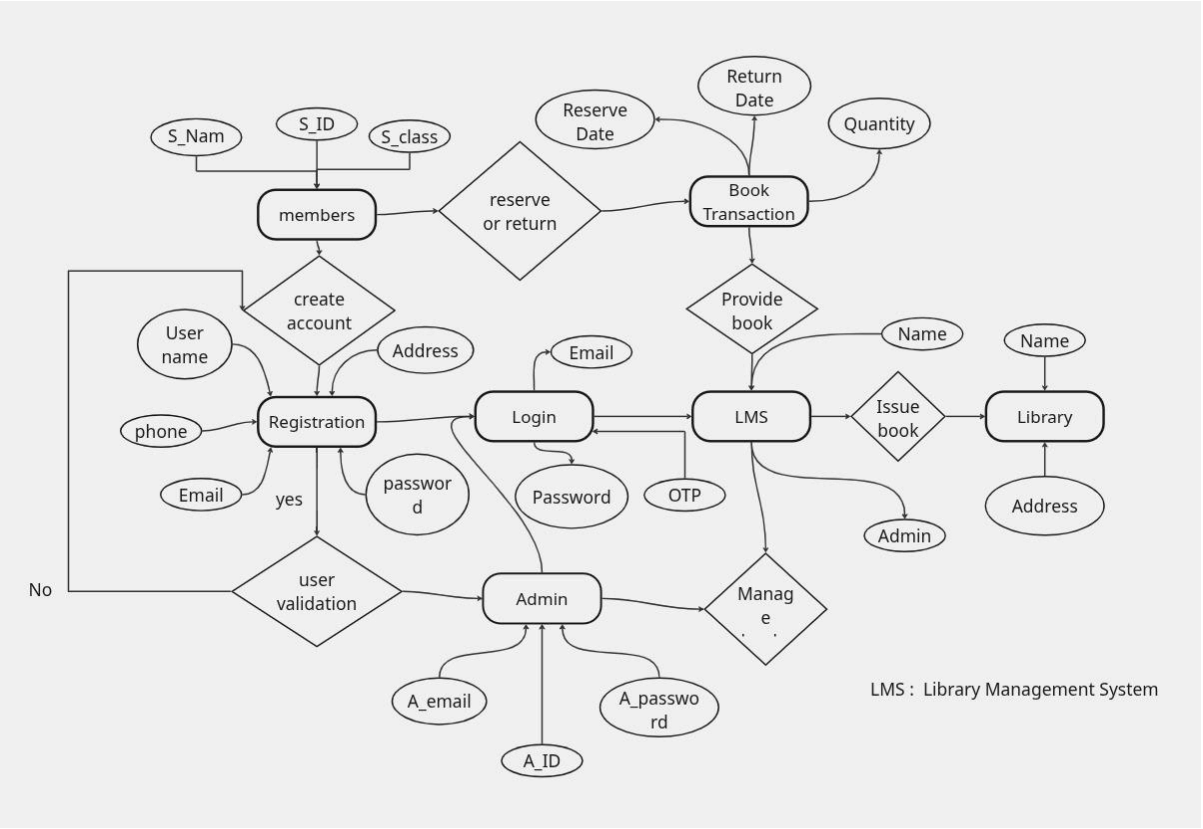

Figure 3.4: E-R diagram for Seminar Library Management System.

Activity Diagram Design

The activity diagram illustrates the progression from one step to the next.The activity could be characterized by a system operation.The flow is drowned from one step to the next. Sequential, branched, or simultaneous progress are all possible for this flow.Activity diagrams include a variety of features, such as join and fork, to show the flow's direction.

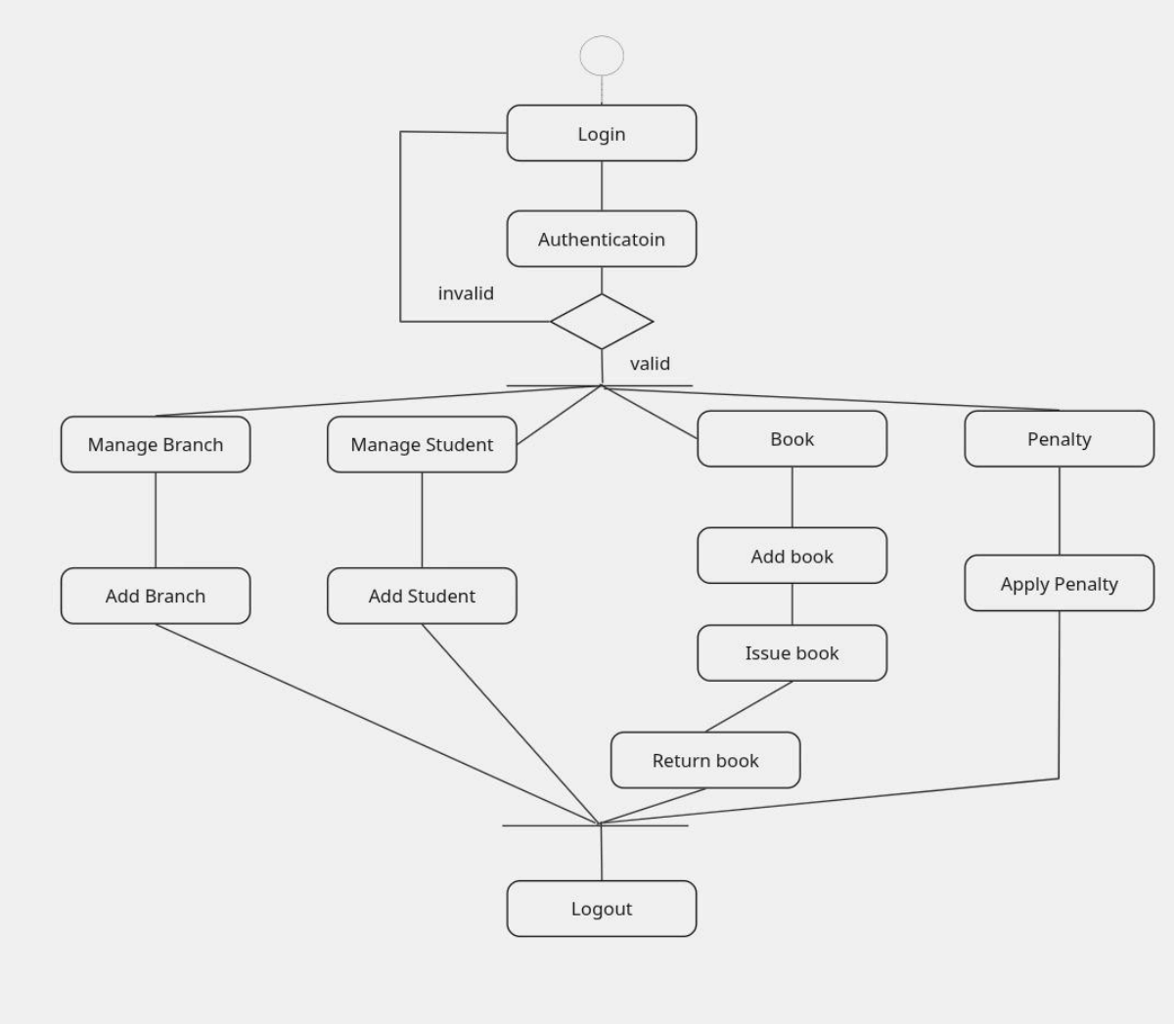

figure 3.5: Activity Diagram for library management system

3.5 System Implementation

During this procedure, the information system's appearance, functionality, and compliance with quality standards are all determined. It covers the system's construction, functionality, and quality assurance.

Software Requirements: The following software will be used to complete the project:

Table 3.1: Software Requirement of the system

Operating system	Linux, Windows XP or later
Browser	Google Chrome, Firefox etc.
Editor	Intellij IDEA, Visual Studio, Sublime etc.

Hardware Requirements: The following hardware will be used to complete the project:

Table 3.2: Hardware Requirements of the System

Processor	Standard Processor
RAM	2GB RAM or more
Hard Disk	50 GB or more
Monitor	Standard monitor
Keyboard	Standard keyboard
Mouse	Standard mouse

3.6 Technologies and Tools

Java stack will be used in this significant project. The technologies will be used such as java, spring boot, HTML, CSS, Javascript, React, PostgreSQL, REST API's etc.

HTML: HTML, or Hypertext Markup Language, is the fundamental building block of the World Wide Web. Its primary function is to define the importance and arrangement of online content. In contrast, CSS caters to the visual presentation of web pages, whereas JavaScript enhances their functionality and behavior. The term "Hypertext" refers to the interconnected links that connect web pages, whether they exist within a single website or span multiple sites. These links are an essential component of the internet. When you share content on the internet and link it to pages created by other users, you are actively participating in the vast realm of the World Wide Web. [15].

CSS: CSS (Cascading Style Sheets) is a simple method for improving the visual aspects of web documents, such as fonts, colors, and spacing [16]. This stylesheet language is used to define the looks of HTML or XML documents by specifying how elements should be displayed across different mediums such as screens, print, speech, and other forms of media [17].

Bootstrap: Bootstrap is an open-source CSS framework for developing responsive, mobile-first front-end web applications. It provides CSS-based design templates for components such as typography, forms, buttons, navigation, and various interface components, with the capability to include JavaScript [18].

JavaScript: JavaScript is a programming language used to make websites more interactive. It is compatible with both the user's device and the web server. It transforms static web pages into dynamic ones, enhancing the user experience. JavaScript is most commonly found in web applications and browsers, but it can also be found in software, servers, and hardware controls. Users can interact with web pages by performing actions such as clicking

buttons, changing colors, displaying images, setting timers, running animations, and using dropdown menus. In essence, while HTML and CSS shape the structure and appearance of a webpage, JavaScript adds interactive features that engage the user. [19]

React: React is an open-source JavaScript library that can be used to create user interfaces (UI) and UI components. It uses a declarative and component-based approach, making it a flexible essential tool for developing single-page applications. React, which was first introduced by Facebook in 2013, is a powerful JavaScript library for building revolutionary applications. It is skilled at managing the interface layer and enabling the development of both web and mobile applications [20].

Spring Boot: Spring Boot is a popular framework for building Java-based applications, particularly microservices, with minimal configuration and maximum efficiency. It simplifies the development process by providing pre-configured templates, auto-configuration, and embedded servers, such as Tomcat or Jetty, to eliminate the need for complex setup. Spring Boot's convention-over-configuration approach allows developers to focus on writing business logic rather than boilerplate code. It seamlessly integrates with the broader Spring ecosystem, offering support for dependency injection, data access, security, and more. With features like starter dependencies, actuator endpoints for monitoring, and seamless cloud deployment capabilities, Spring Boot is a powerful choice for modern application development.

PostgreSQL: PostgreSQL is a robust, open-source relational database management system (RDBMS) known for its reliability, extensibility, and compliance with SQL standards. It supports advanced data types, full-text search, and powerful indexing methods, making it suitable for handling complex queries and large datasets. PostgreSQL's extensibility allows developers to define custom functions, data types, and operators, enabling the creation of specialized solutions. Additionally, its support for ACID (Atomicity, Consistency, Isolation, Durability) compliance ensures data integrity and reliability. PostgreSQL is widely used in a variety of domains, including web applications, data analytics, and geographic information systems (GIS), thanks to its scalability, performance, and rich feature set.

REST API: A REST API (Representational State Transfer Application Programming Interface) is a widely used architectural style for designing networked applications. It enables communication between client and server over HTTP, using a set of stateless operations. REST APIs rely on standard HTTP methods—such as GET, POST, PUT, DELETE—to perform actions on resources, which are typically represented as URLs. The responses are commonly formatted in JSON or XML, making them lightweight and easily consumable. REST's simplicity, scalability, and flexibility make it a preferred choice for building web services, as it allows different systems to interact seamlessly while maintaining independence in their implementation.

3.7 Testing and Maintenance

Finding errors is the main objective of testing. Testing entails making an effort to find every potential flaw in our creations. It's comparable to making sure software performs as intended and doesn't malfunction negatively. Software testing includes verification and validation. Software testing can be done in two ways.

Black box or functionality testing: In this type of testing, testers make sure the program functions correctly on the outside but have no understanding how it is made internally. It ultimately boils down to figuring out how the software works and behaves. White box testing, sometimes referred to as implementation testing, is a type of structural testing that is used to assess the internal operations, circulation, and structure of software. In order to verify that the programs' routes and logic flows satisfy the requirements, the tester develops test cases during this phase of testing.[24]

Test goals:

All field data should function correctly.

I. Links ought to take users to the desired pages.
II. Furthermore, the input screen, messages, and opinions must all be uninterrupted.
III. Features that will be examined
IV. Make sure every entry is formatted appropriately.
V. Avoid re-entering.
VI. Make sure users are sent to the appropriate pages by each link.

Chapter 4

RESULTS AND DISCUSSION

During the course of this project, I develop a library management system that offers an optimal user interface, simplifying tasks for both librarians and students. Utilizing this platform streamlines book and student maintenance processes. Particularly, scanning QR codes from book covers enhances the efficiency of library transactions for librarians. Moreover, this system ensures top-notch security and additional services for all users.

4.1 Outlook of the System

Following are the Outlook of working website which I have developed:

i) Homepage:

The website's home page is displayed to the user when they enter the URL. The user can view the list of books with the book ID format number along with some further details and the quantity of copies that are available from this page. In addition, the homepage area contains all of the most recent notices. Both the admin and the user must log into the system to access their respective sections after finishing the registration process. The homepage provides information about the library hour.

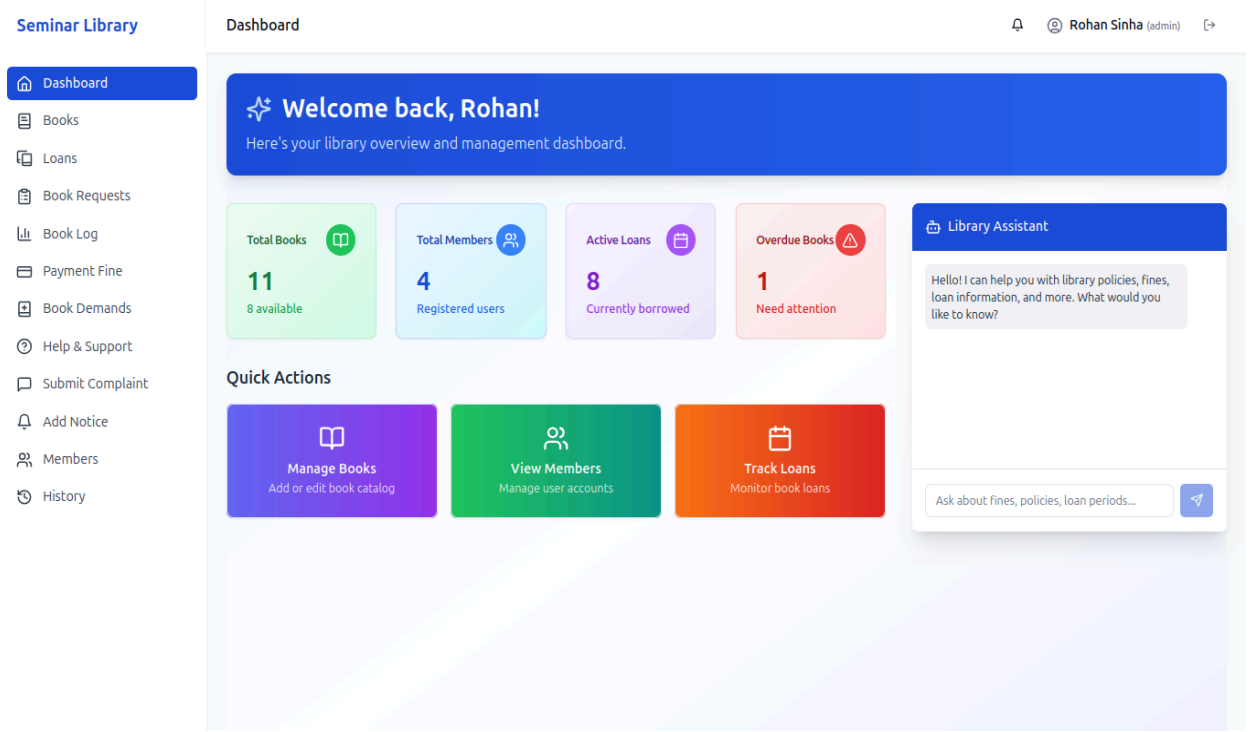

Figure 4. 1: Home page for the Library Management System

ii) List of Available Books

This page displays the list of available books stored within a database. Each book entry is presented with a structured table format showcasing specific description such as the book's ID format, books name, book author name, category, edition, total quantity, and the number of copies currently available for borrowings.

To make sure manageable viewing, the page is designed to display up to 13 books per page. This organized presentation allow users to easily browse through the collection, facilitating efficient direction and selection of desired reading materials.

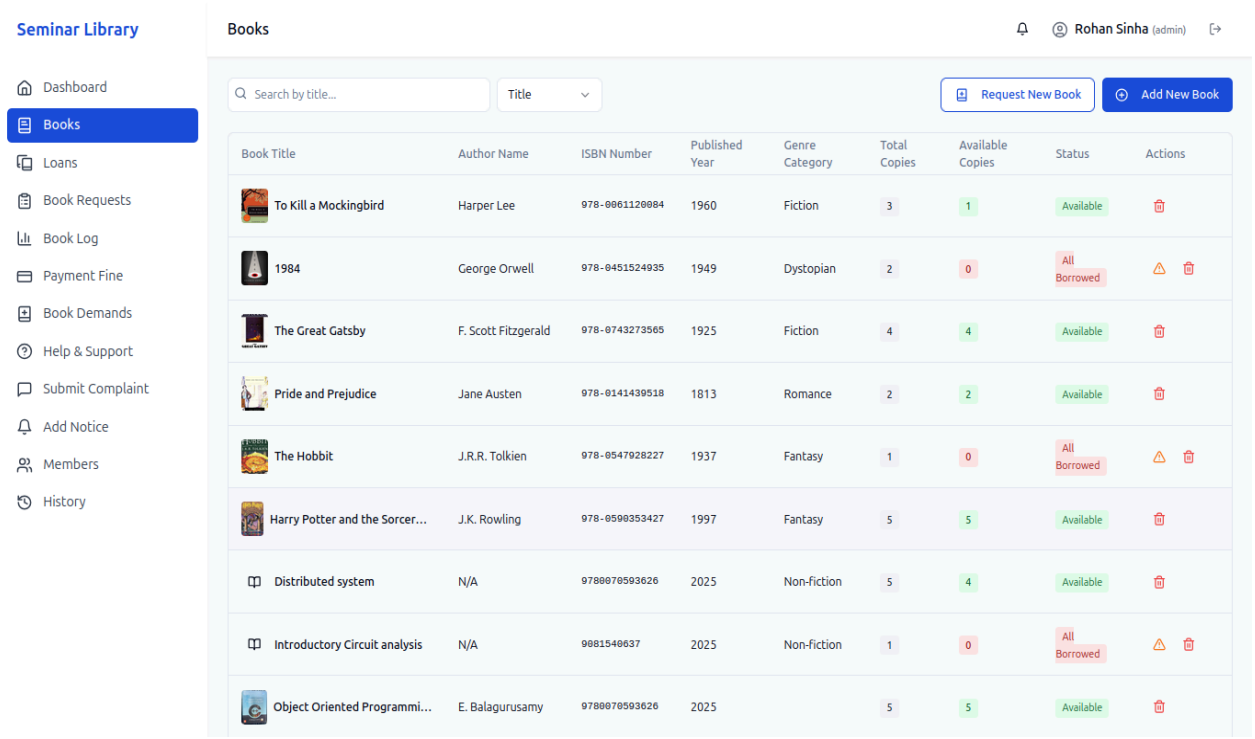

Figure 4. 2: Available all book list

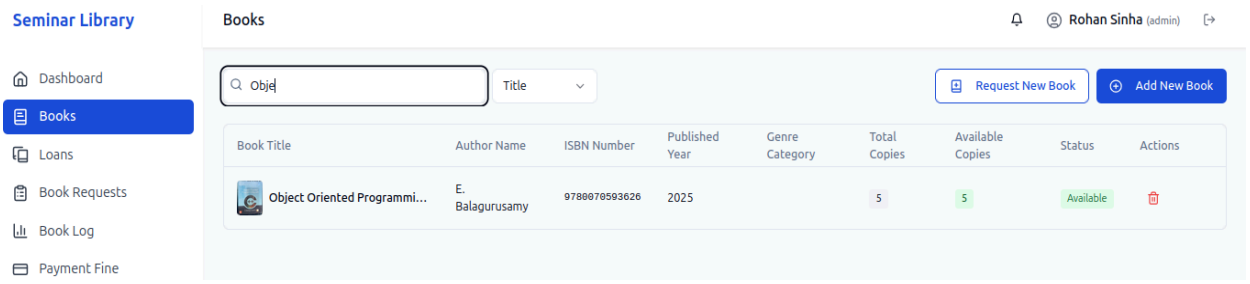

Figure 4. 3: Search Result

iii) Top requested book

In this section the user or admin can see which is the top requested book and according to it admin can increase the number of copies of the book in the library.

Figure 4.4 : Top requested book

A. Admin Section:

i) Admin Registration and Login Page:

The admin has his own distinct registration and login page. Librarians are need to fill out a registration form with all the necessary information. Once registered, admins can seamlessly access their dedicated sections using their authorized username and password. If either the username or password is incorrect, access to the system is denied. The registration form includes vital information such as admin name, username, email, password, phone number and address.

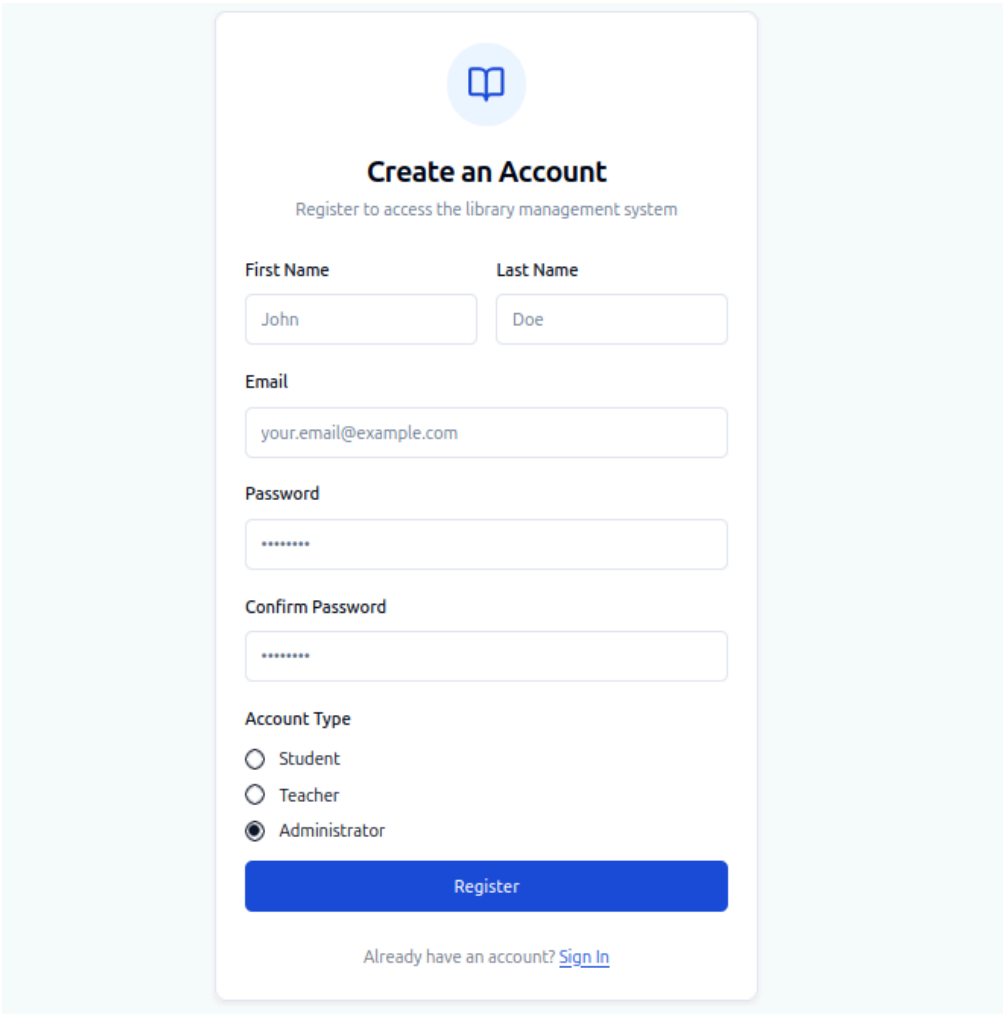

Figure 4. 5: Admin Registration Form

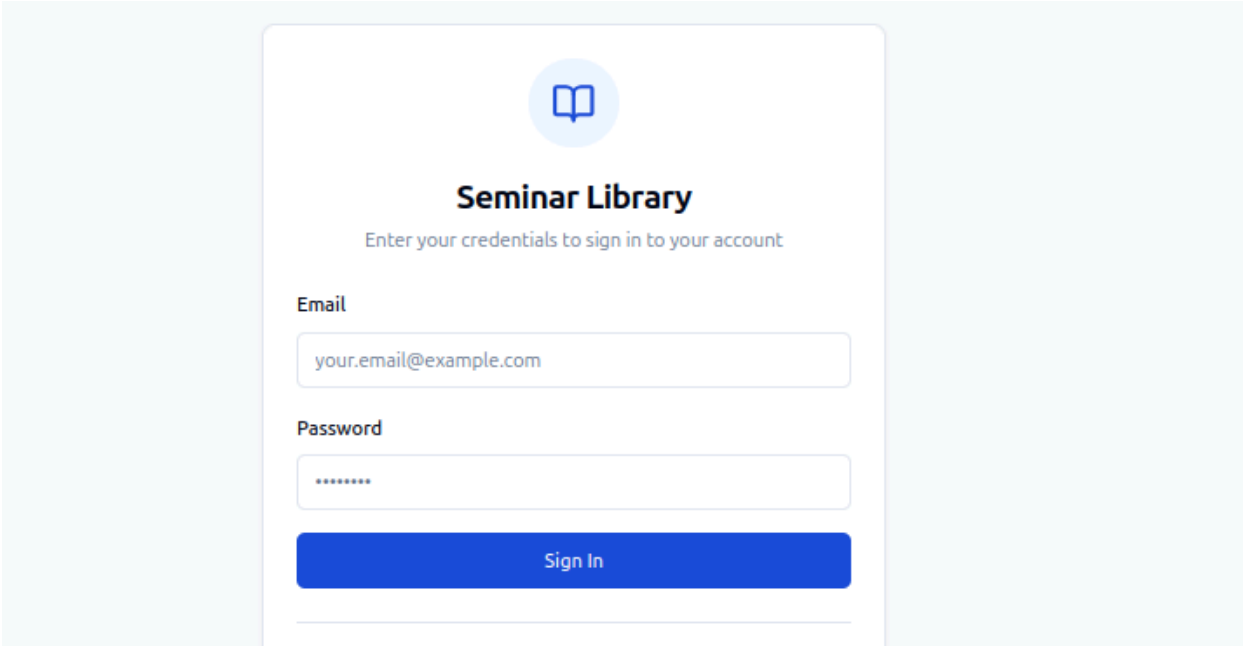

Figure 4. 6: Admin login form

ii) Admin dashboard

After logging into the admin panel, an extensive overview page will appear, giving admin a quick glance at important data. Here, the admin gains insights into various aspects such as the total number of members, the books in the collection, the total issued books, accrued fines, and book requests, empowering them with the necessary information to efficiently manage the library system.

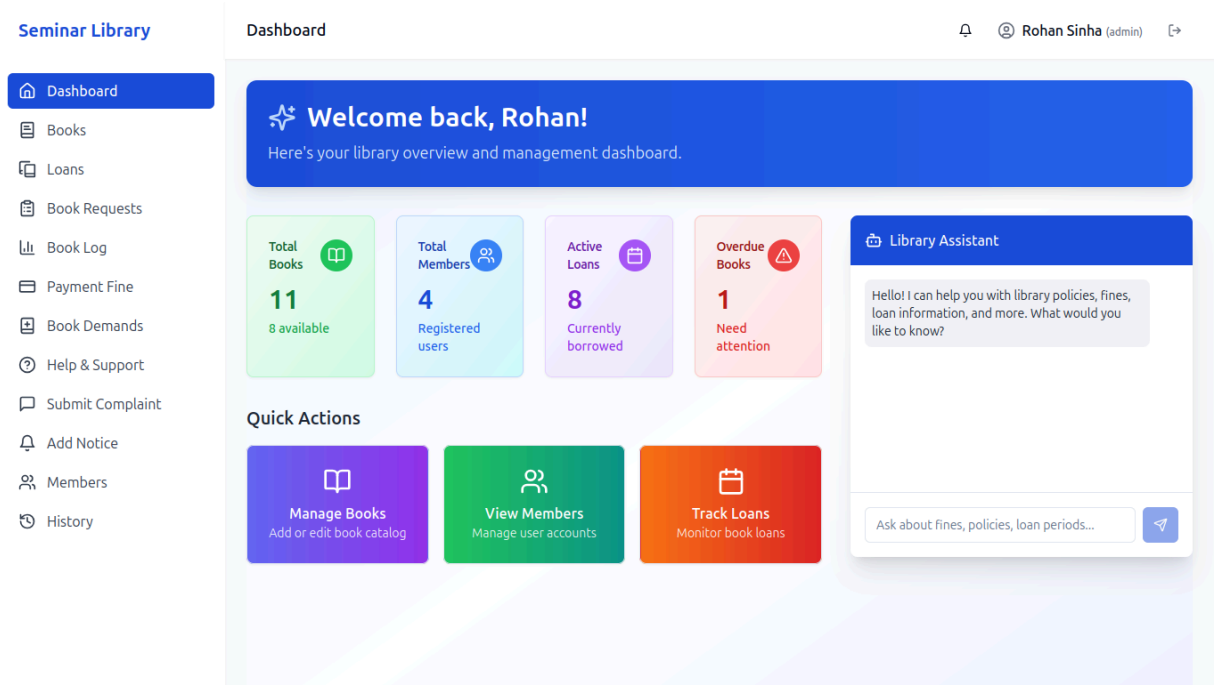

Figure 4. 7: Admin Dashboard

iii) User Status:

On this page, all requests for access to the system are presented in a tabular format for easy management. Each request includes some information such as the user's name, username, and email address. Initially, the status of each request is marked as "pending." Additionally, two buttons are provided for an action: one for approval and another for decline.

When the admin hits the "approve" button, the status of the request changes to "yes," indicating approval of access. Conversely, if the admin chooses to decline the request, the status is updated to "no," signifying denial of access. This systematic approach simplifies the process of reviewing and responding to user access requests, ensuring efficient management of system permissions while maintaining clarity and transparency in decision-making.

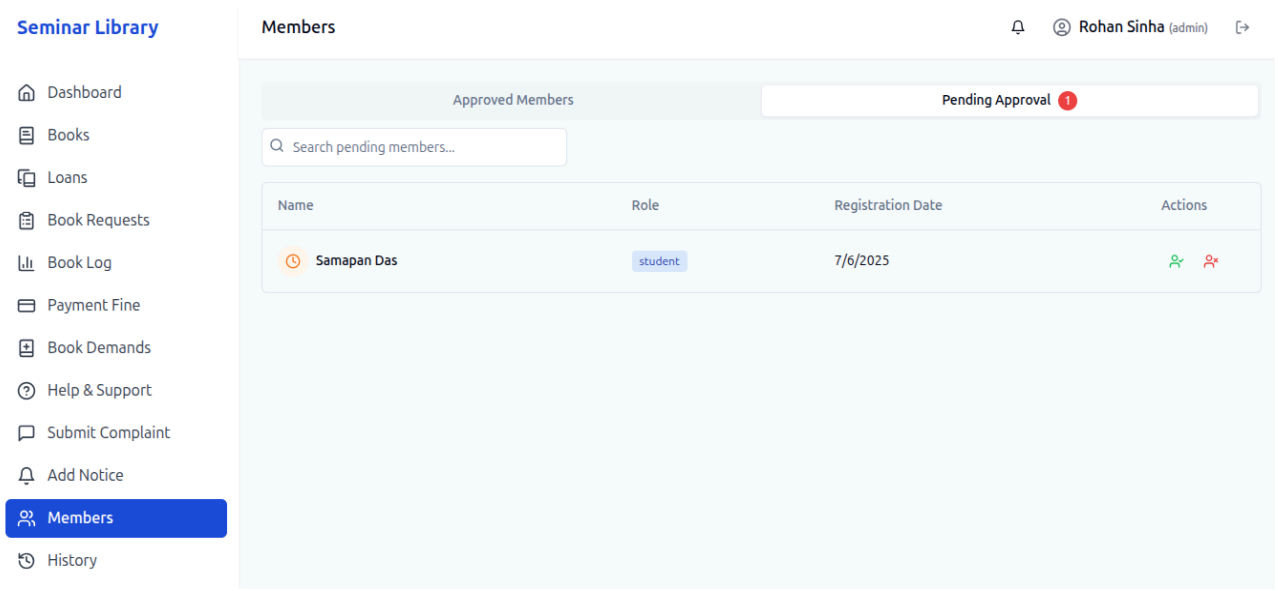

Figure 4.8 : User Status Page

iv) All approved user's Information

On this two-page, librarians will find a comprehensive display of all registered teachers and students, showing the information they provided during the registration process. This wealth of information serves as a valuable resource, enabling the librarian to easily contact individuals if needed. He can also search for a particular member or students if necessary.

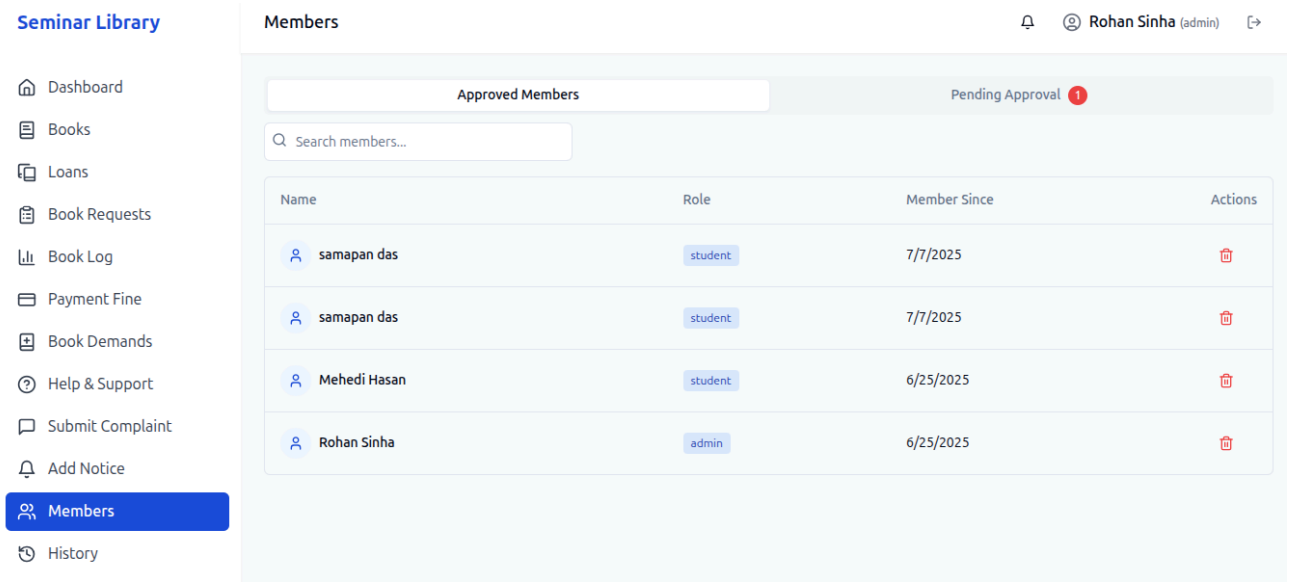

Figure 4. 9: Approved Members Information

v) Add Book

Admin must complete the form that captures every detail of a book in order to add new books to our library database. The form includes book's name, books author's name, books

image, books digital file (if available), books publication, books ID, book edition, books category, total copy and available copy. The book's ID needs to be unique. The admin can see the books list with detailed information in the system.

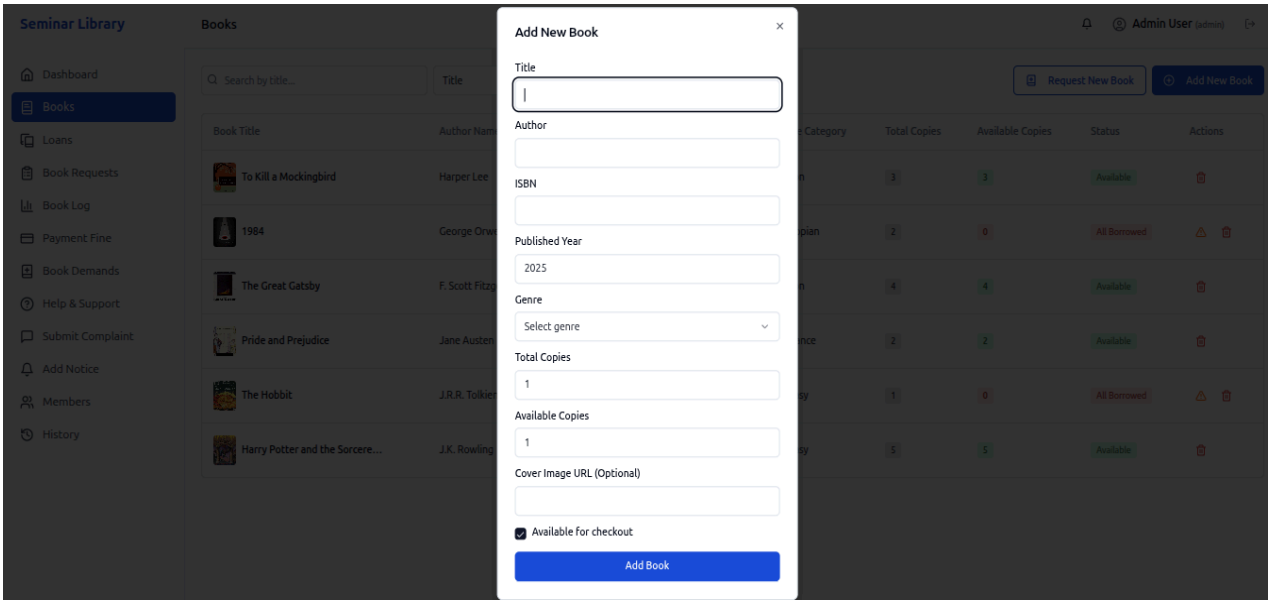

Figure 4. 10: Add Book form

vi) Book request approving

A request from the user for a book and the admin will approve that book with a particular due date.

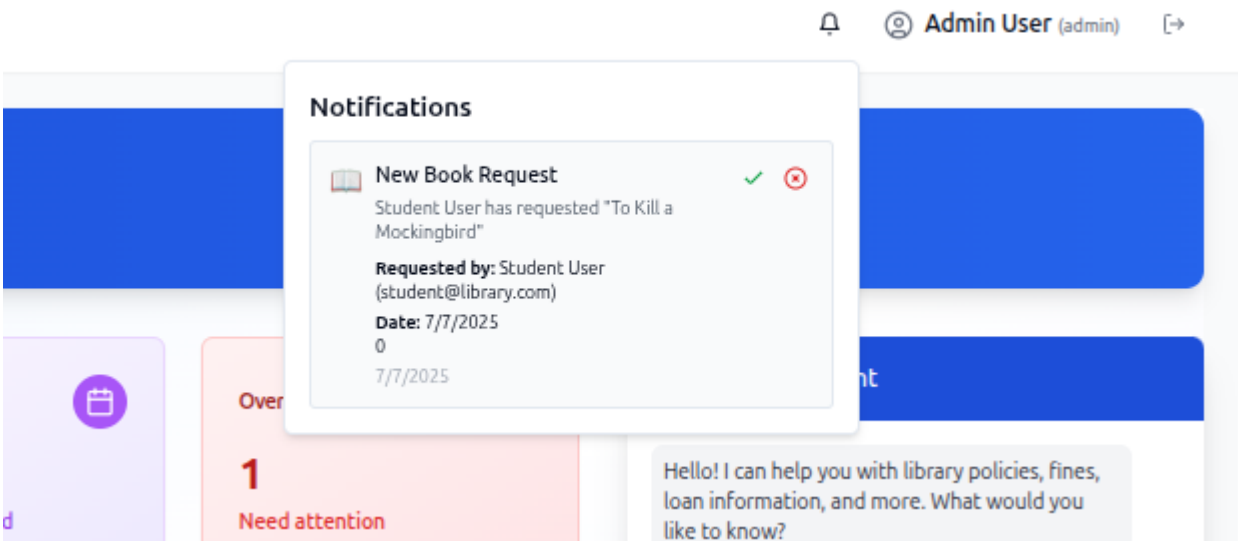

Figure 4.11: Book request notification from user

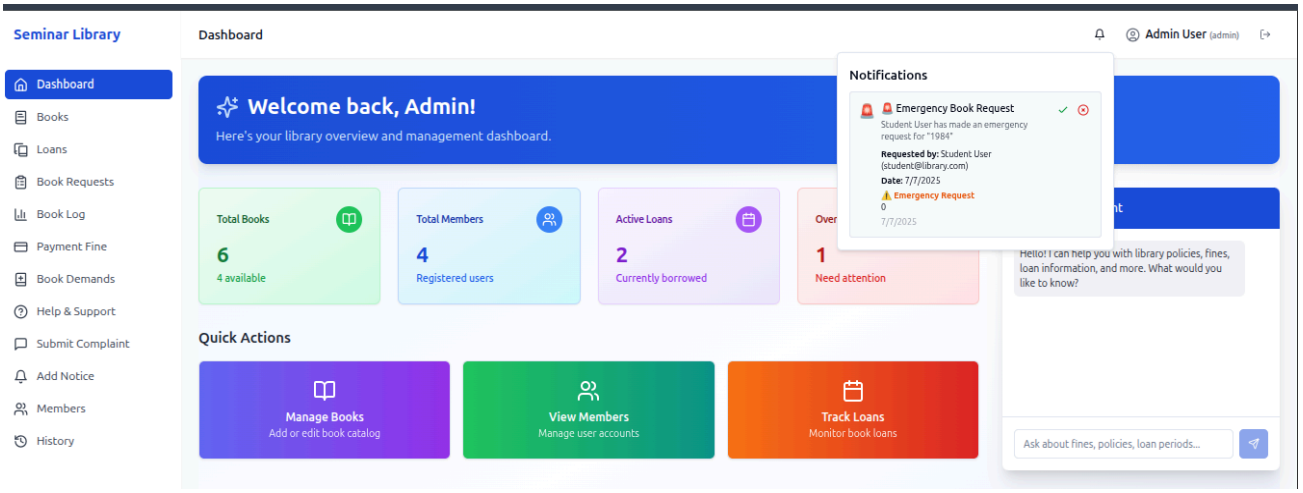

Figure 4.12 : Emergency book request from user

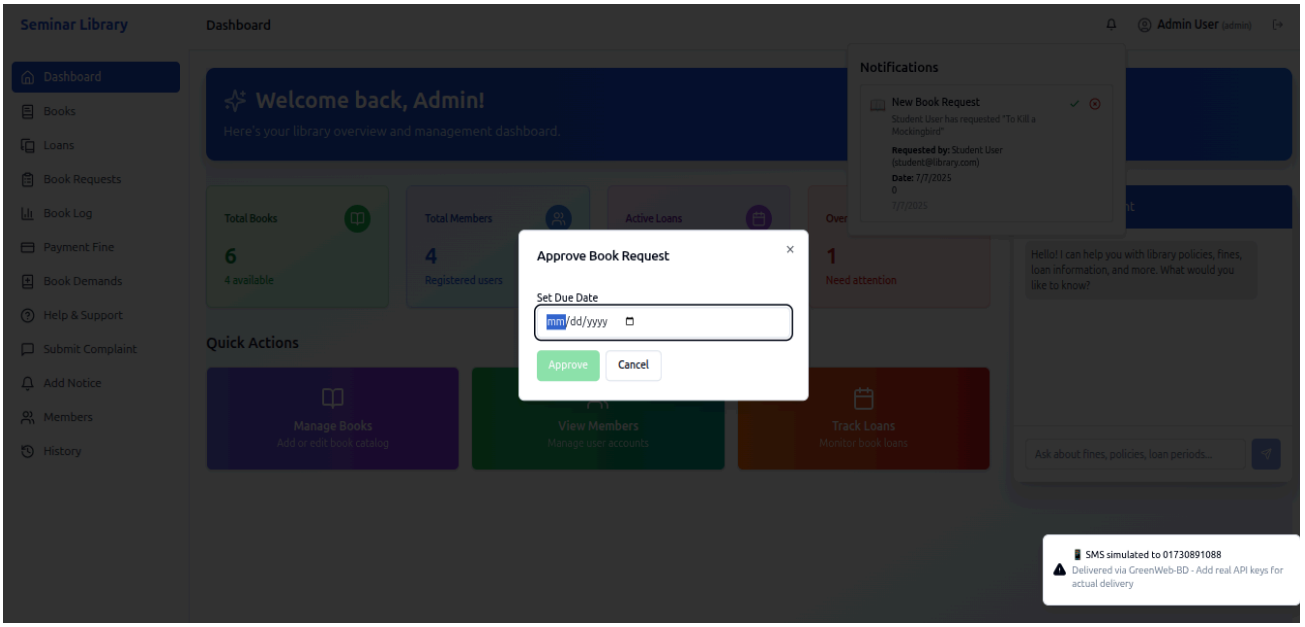

Figure 4. 13: Approving the book request with a due date

vii) **Loan Table**

When it comes time to return a book, the admin needs to first go to the all issued book page. Here he can see the all-necessary information along that book with a button mentioned as “Return” button. After pressing the “Return” button from the issued books list, the admin can return the book from that particular student. There is another button mentioned as “Delete” button for the deletion of the user information who took that book.

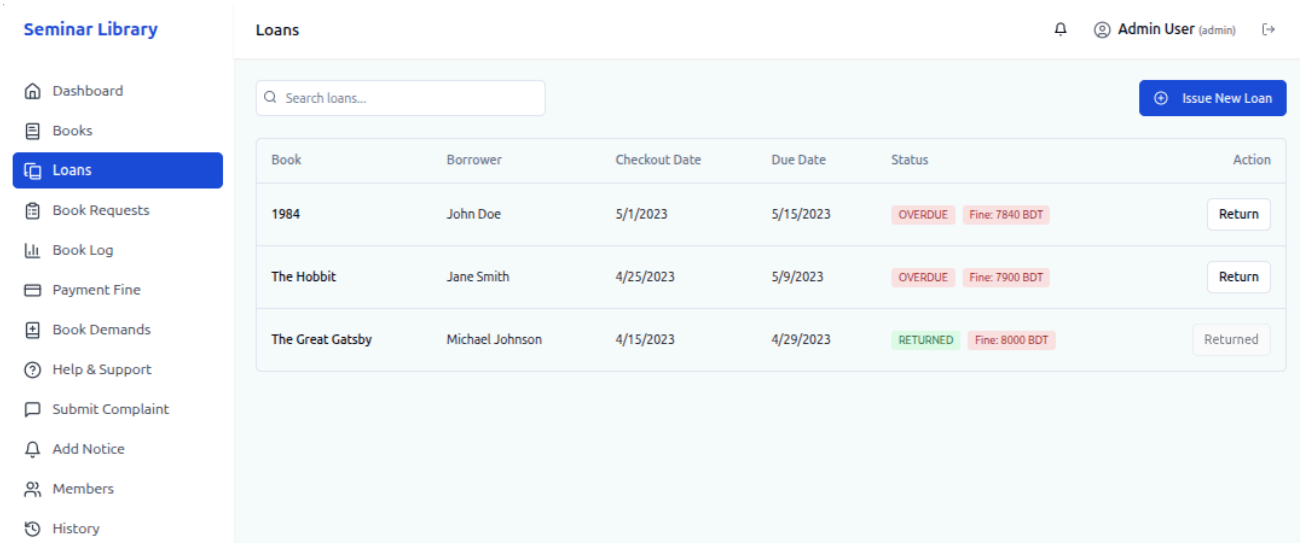

Figure 4. 14: Loan Table

viii) Requested Books List

This page displays the books that the teachers and students have requested. Students and teachers can request books that they require for the purposes of their studies. The request will be presented to the admin in tabular form, with the user's name, email address, book name, book author name and book publication name displayed.

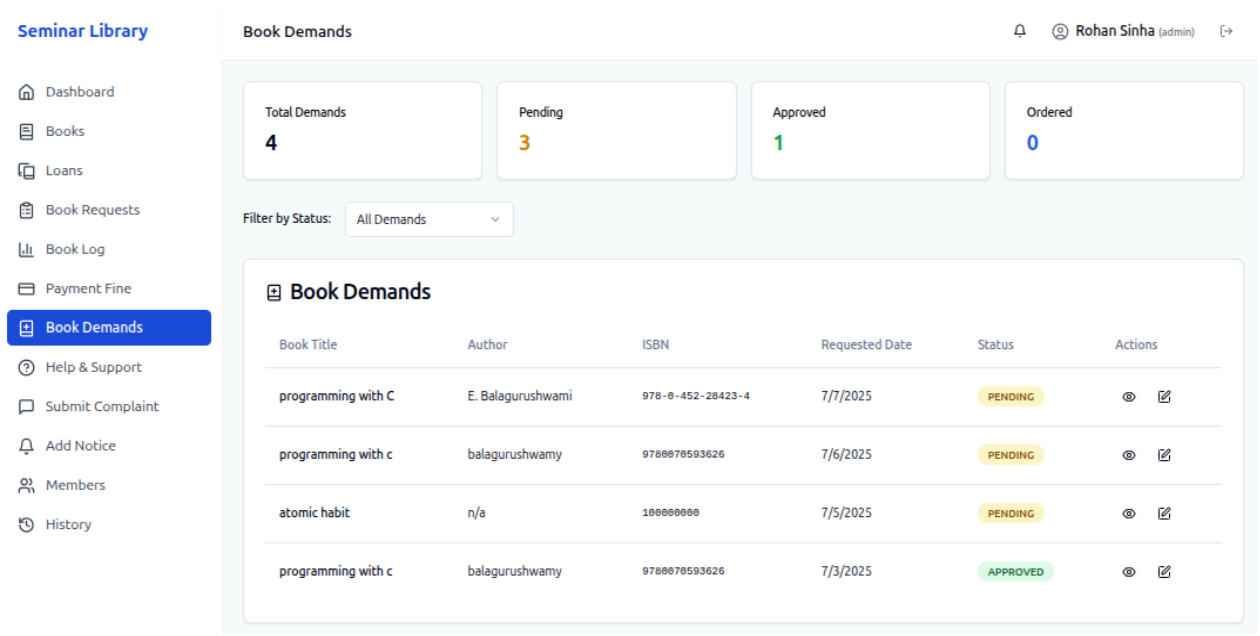

Figure 4.15 : Requested Books table

ix) Fine and Fine payment

In this page, when a user fail to return a book on time, their relevant details are displayed including their name, email address, the name of the overdue book, and the amount of fine insights. The goal of this display is to identfiy the administration and the user about the

overdue book and the corresponding fine.

Besides, an option to delete the user information is provided after the fine has been issued. This allows for the removal of the user detail from the system once the fine have been paid or resolved. This feature helps maintain the database accuracy and ensures that records are up to date. The delete button provides an efficient way to manage user information and book returns facilitating the library's operations and maintaining accountability for borrowed items

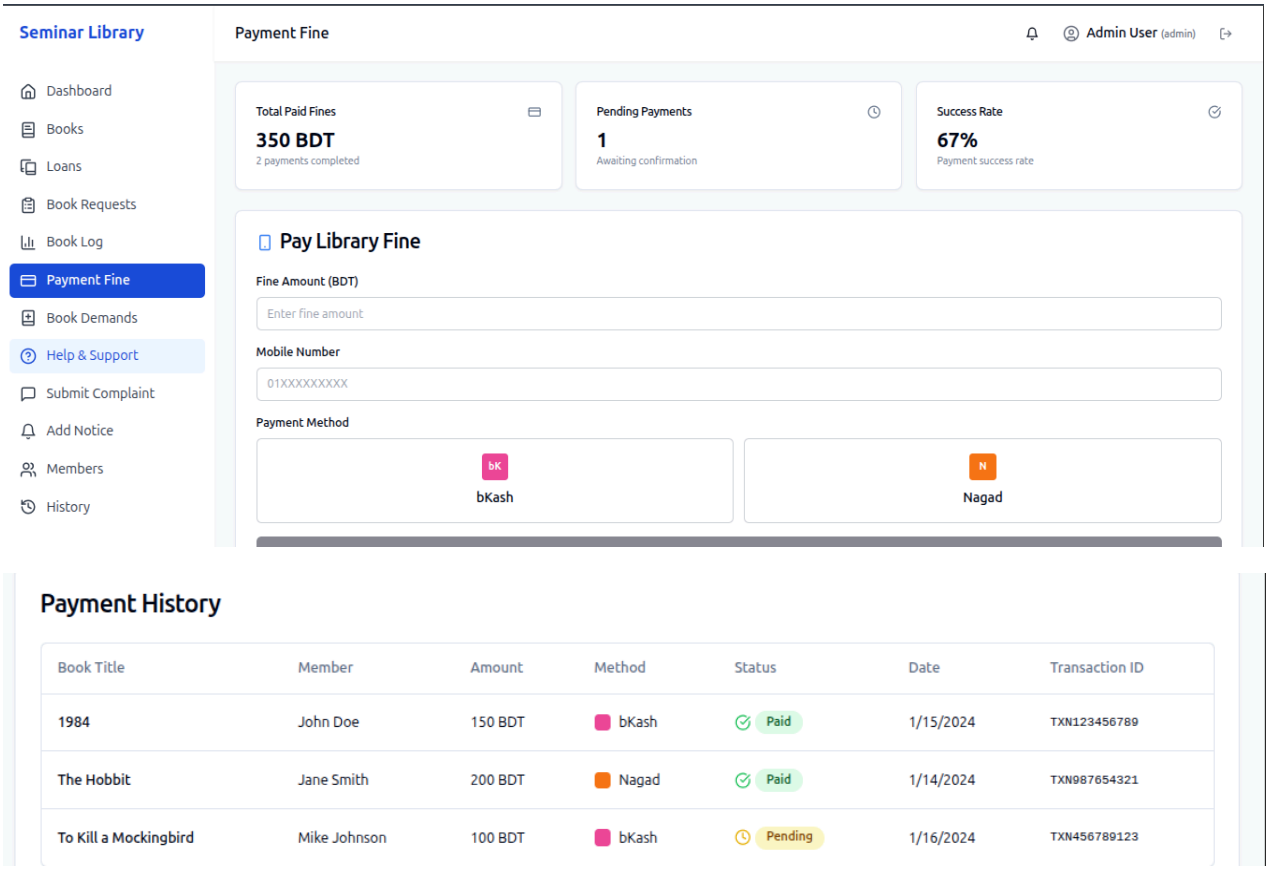

Figure 4.16: Fine payment section

x) Add Notice

The admin has the capability in add important notices to a website through the designated "add notice" section. Once posted, these notice is visible to everyone within the system. This ensures that all members, users, or visitors to the website are informed about the latest detail or updates provided with the admin. Moreover, the admin has the authority in alter or delete these notices as needed. This system allow for efficient communication and dissemination of important information to all the users. These notices can pertain in various updates, announcements, or important information relevant to users.

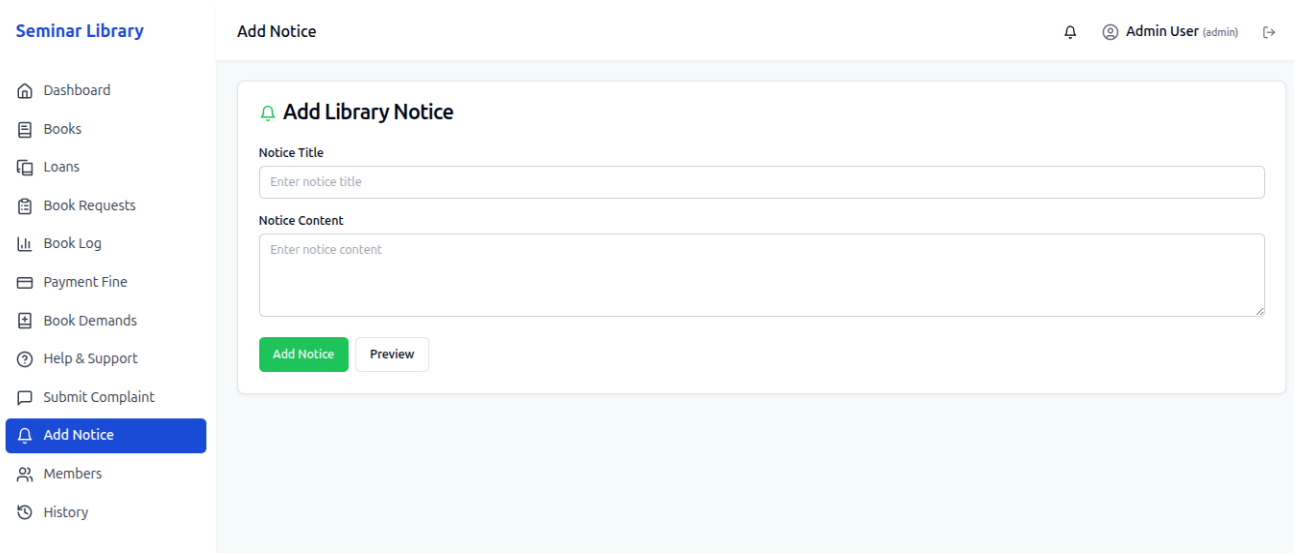

Figure 4.17: Add notice section

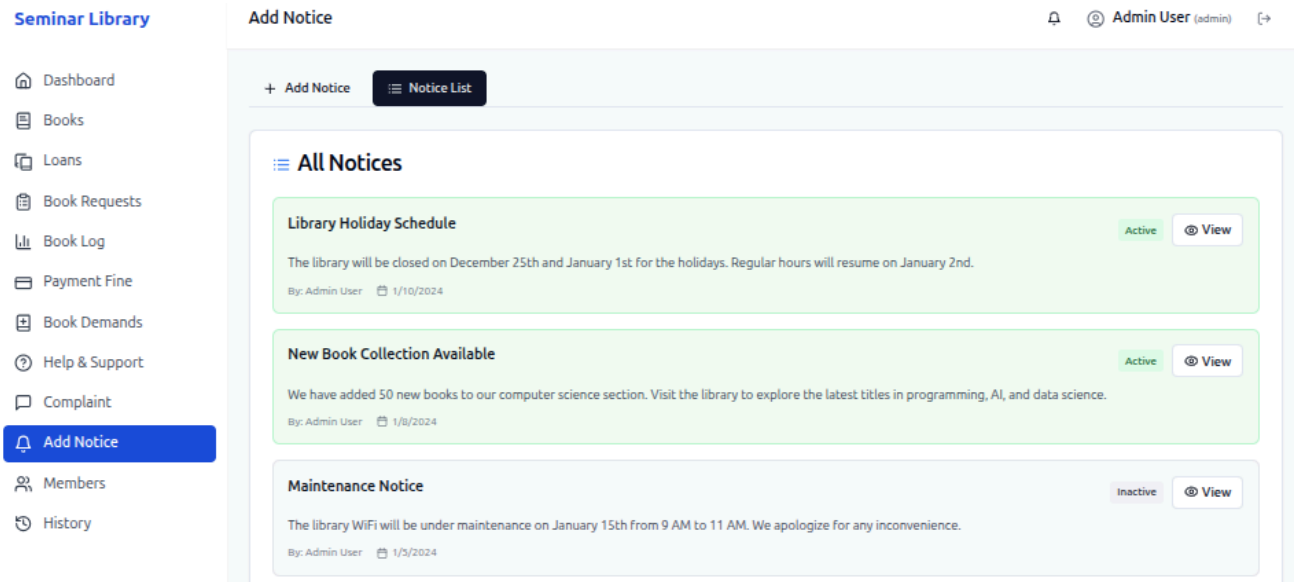

Figure: 4.18: All notice

B. Student Section:

i) Student Registration:

Students have their own distinct registration and login page. Students have to complete a registration form, providing essential information. Once registered, students can seamlessly access their dedicated sections using their authorized username and password. If either the username or password is incorrect, access to the system is denied. The registration form includes vital information such as student name, student username, student email, student password, student phone number, student semester, student department, student roll number and student address.

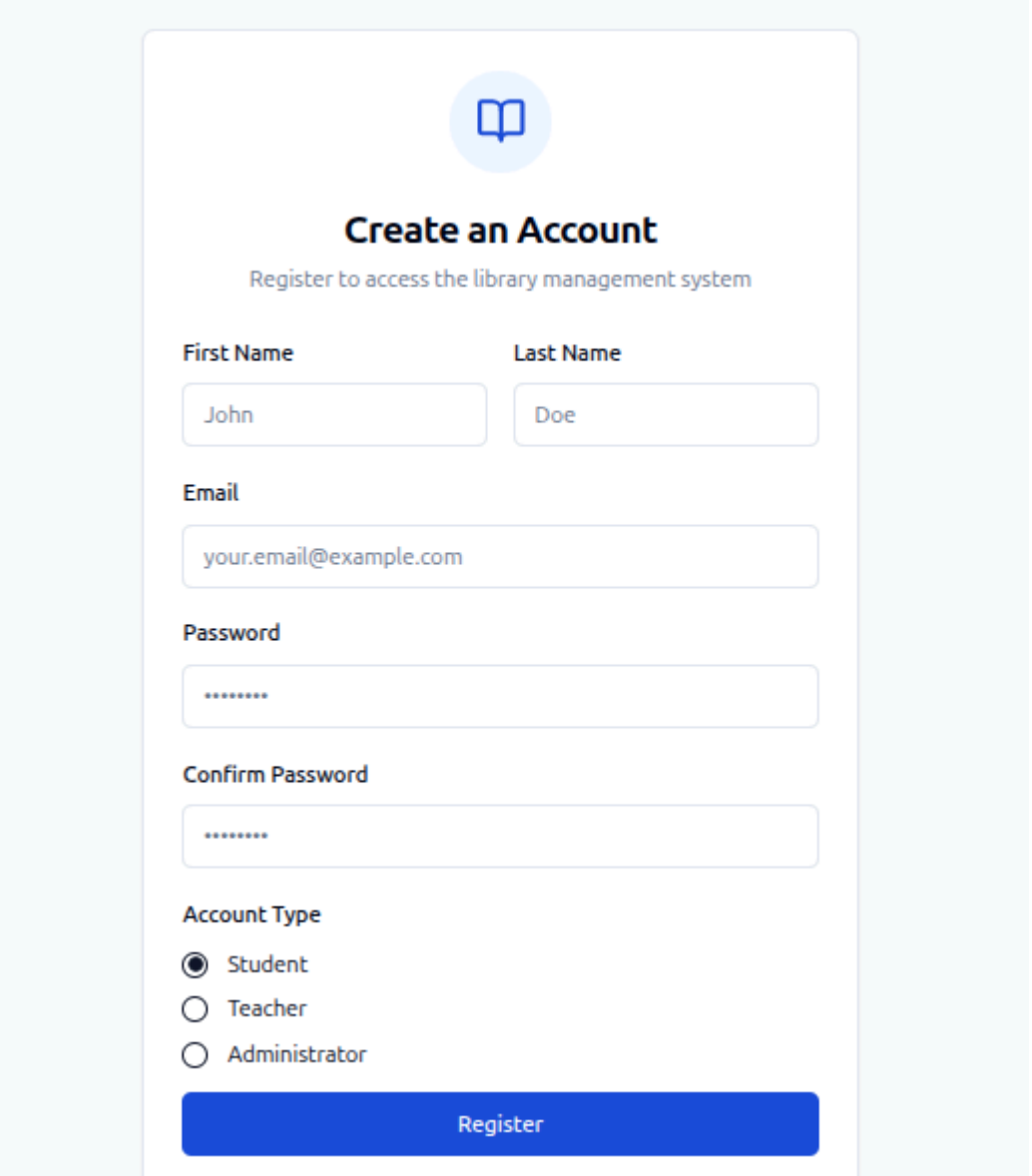

Figure 4.19: Student registration page

ii) Student Dashboard:

Once logged in to the user panel, students are directed to a page displaying their issued book list within their profile. This page presents various details regarding the books, such as issued book titles, dates of issuance, return dates, and other relevant information. Users can conveniently access and manage their borrowed books from this interface, facilitating efficient tracking of their library transactions and ensuring timely returns.

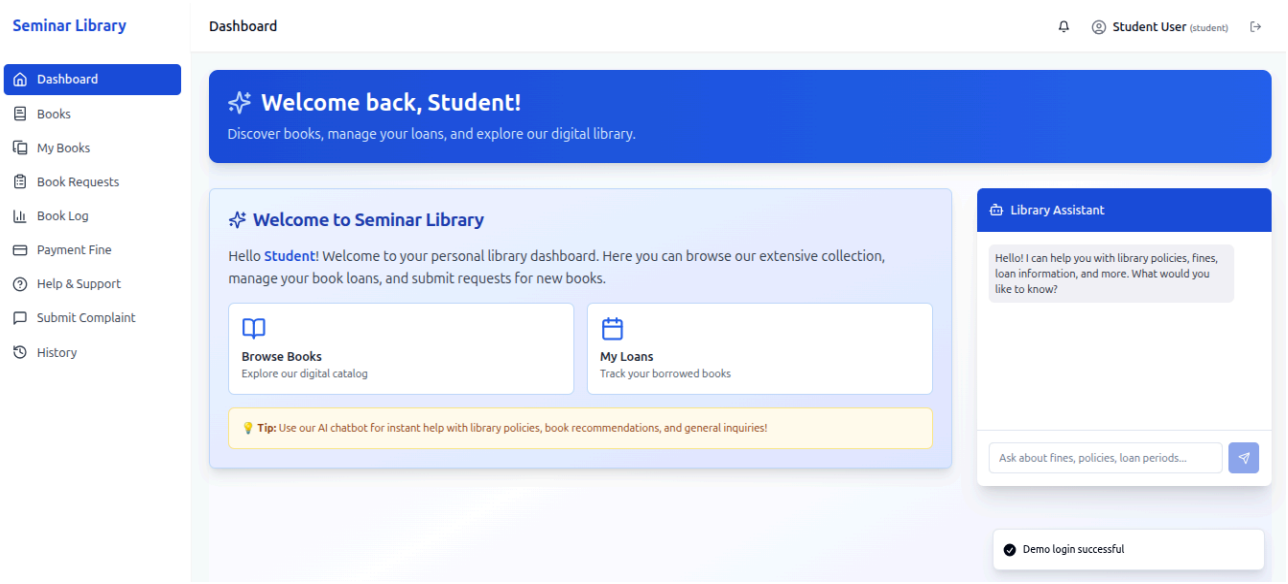

Figure 4.20: Student dashboard

Library Assistant (Chatbot) :

Added a chatbot which will act like a library assistant. Users can ask their queries to understand the library policy and other necessary information.

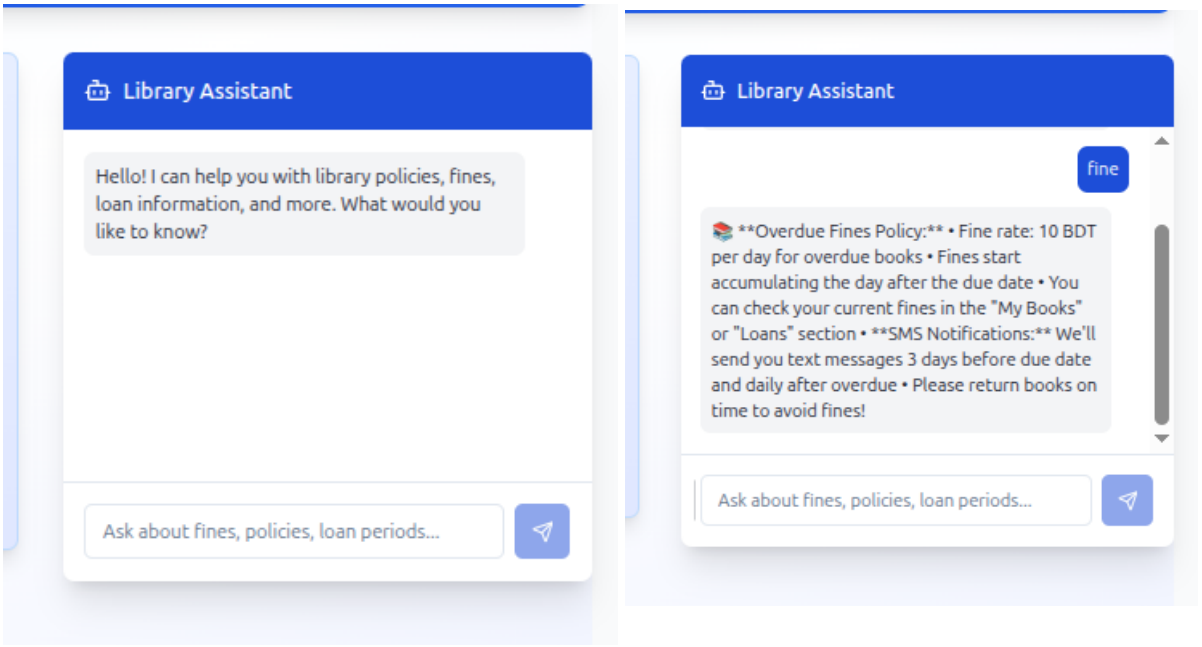

Figure 4.21 : Library Assistant (Chatbot)

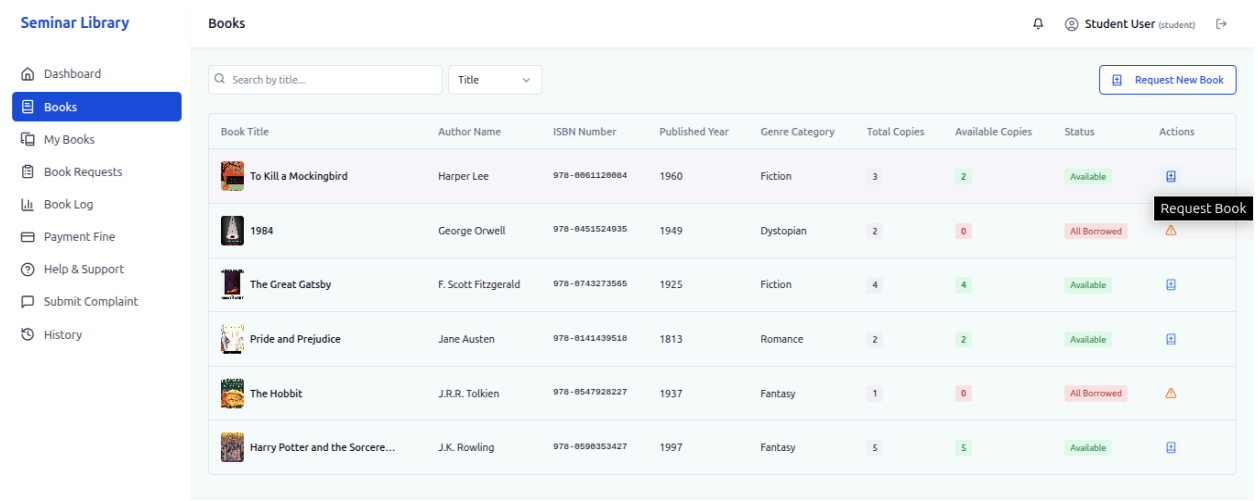

Figure 4.22: request book button

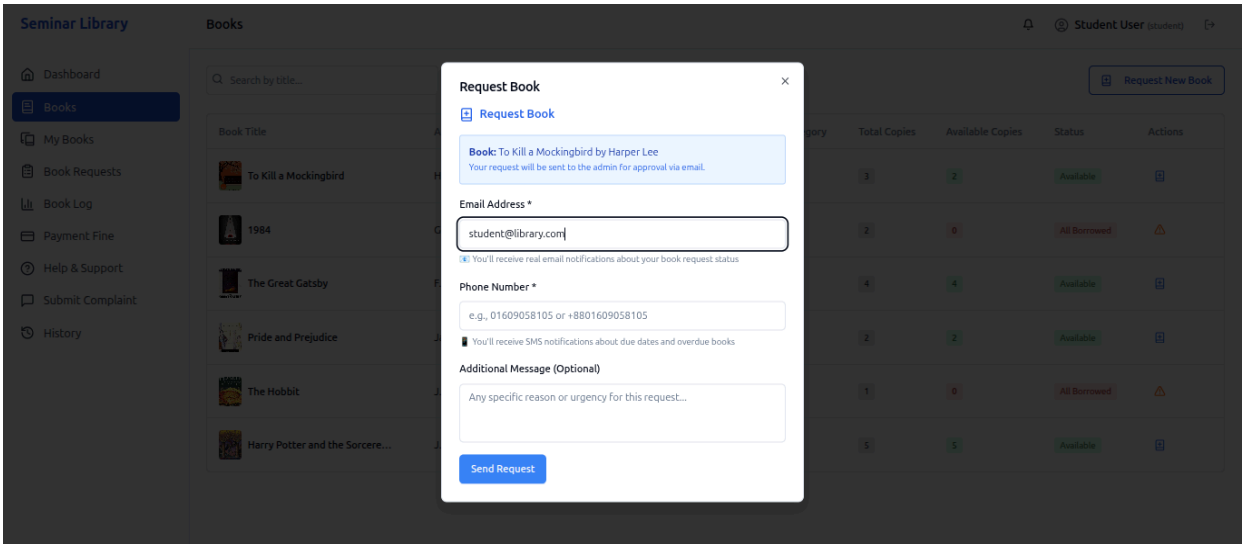

Figure 4.23: request book form

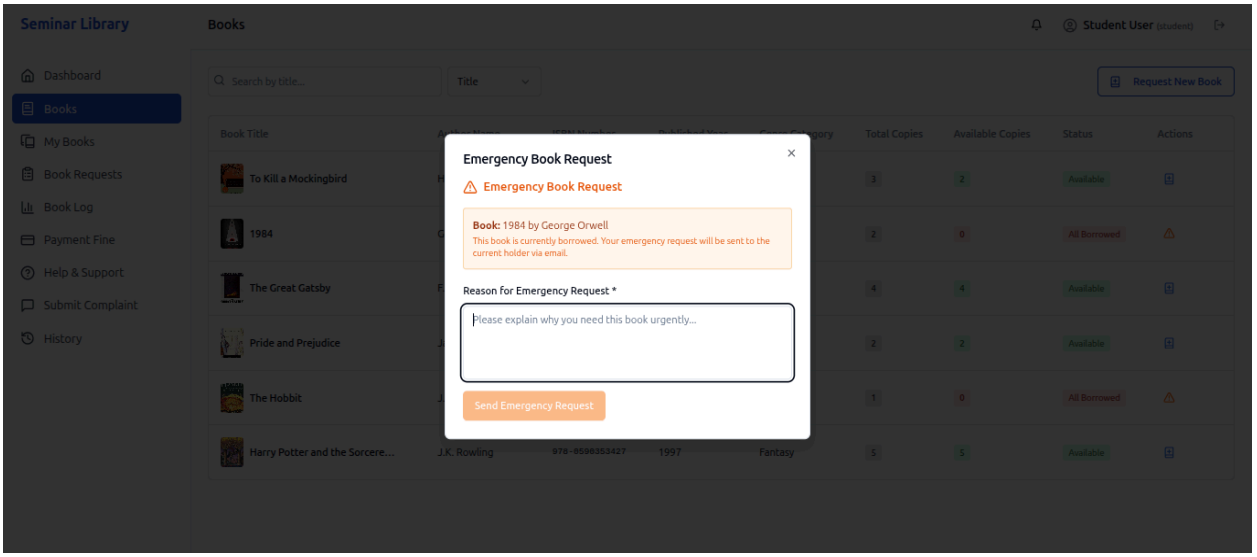

Figure 4.24: Emergency book request form

iii) Borrowed books

In the "Borrowed Books" section, users can view a comprehensive list of all the books they have currently borrowed from the library. This includes important details such as the book title, author, borrow date, due date, and return status. By accessing this section, users can keep track of their borrowing history, manage return deadlines efficiently, and avoid late fees. It serves as a personalized dashboard for users to monitor their library interactions and maintain accountability for the books they have checked out.

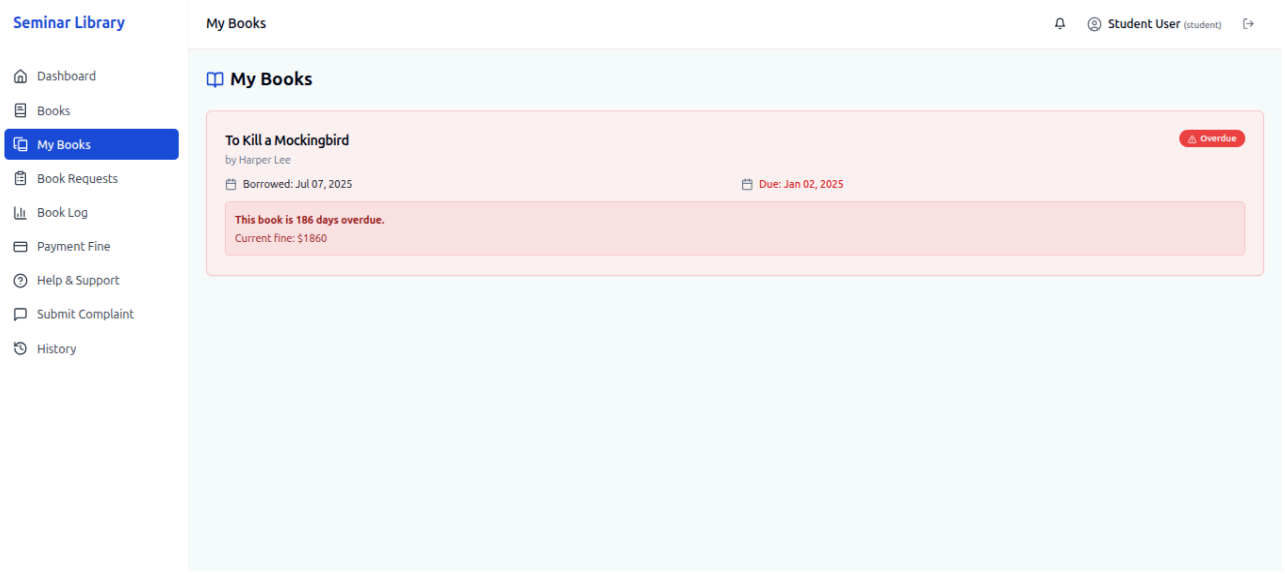

Figure 4.25: User’s borrowed book list including fine

iv) Complaint

The "Complaint Box" allows users to submit any issues, feedback, or concerns they encounter while using the library system. Whether it's a missing book, a technical glitch, or a service-related complaint, users can easily report their problems through this feature. It provides a direct channel of communication between the user and the library administration, ensuring that user grievances are acknowledged and addressed promptly. This helps improve the overall user experience and maintain the quality and reliability of the library services.

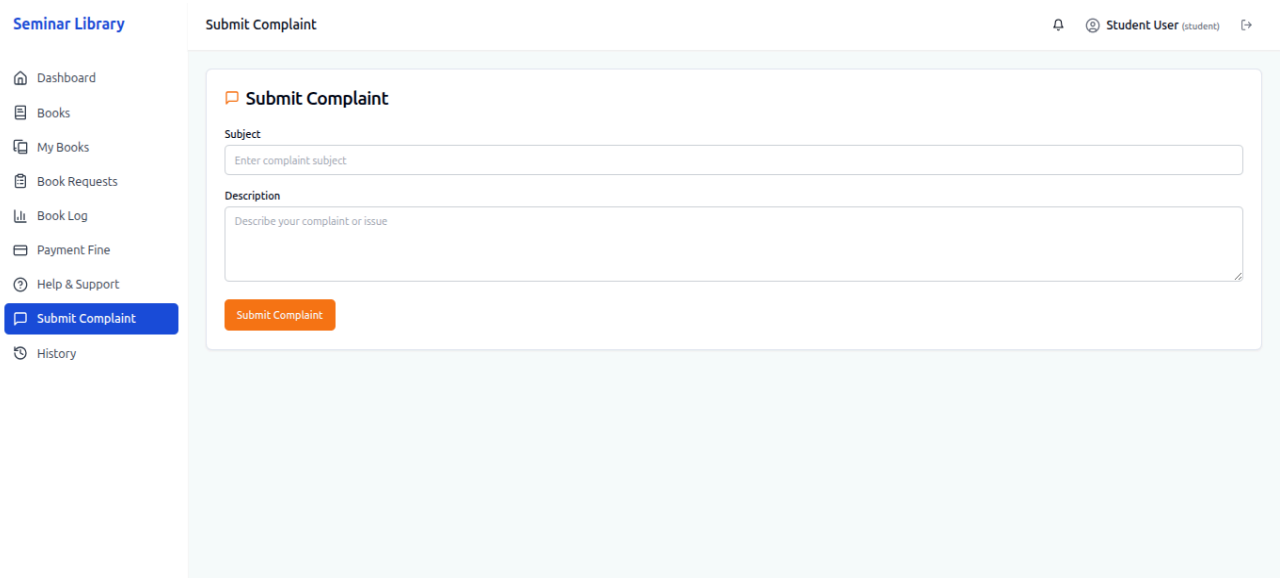

Figure 4.26: Submit Complaint

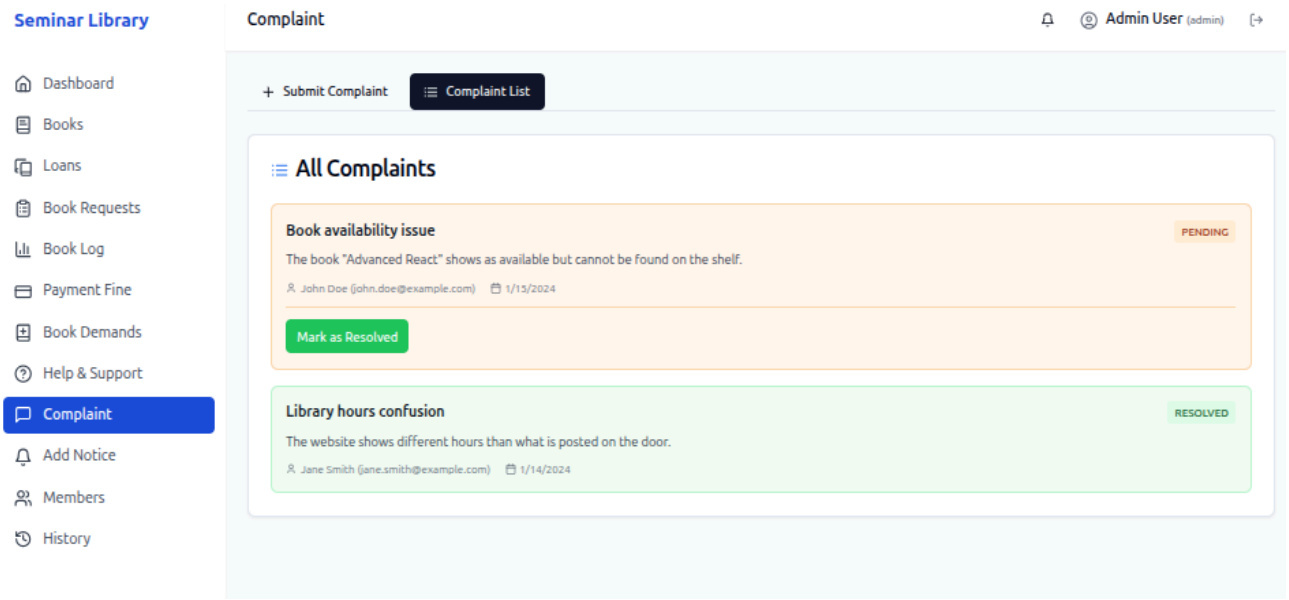

Figure 4.27 : All complain list

v) History

The "History" option provides users with a detailed record of all their past interactions with the library system. This includes previously borrowed and returned books, past fines or payments, and any other completed transactions. By accessing this section, users can review their borrowing habits, check when they borrowed specific books, and track their overall usage of the library. It serves as a useful reference for users to recall past activities and helps maintain transparency in their library account.

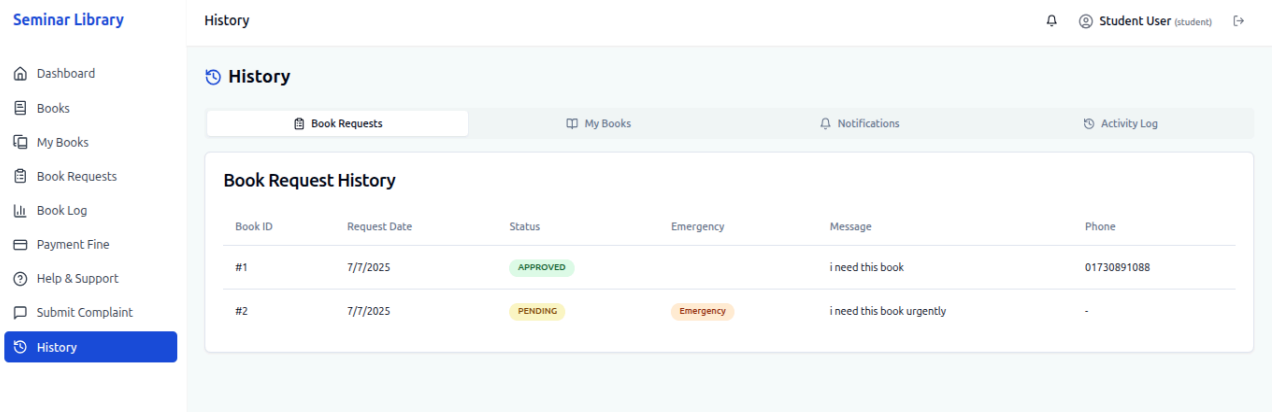

Figure 4.28: book request history

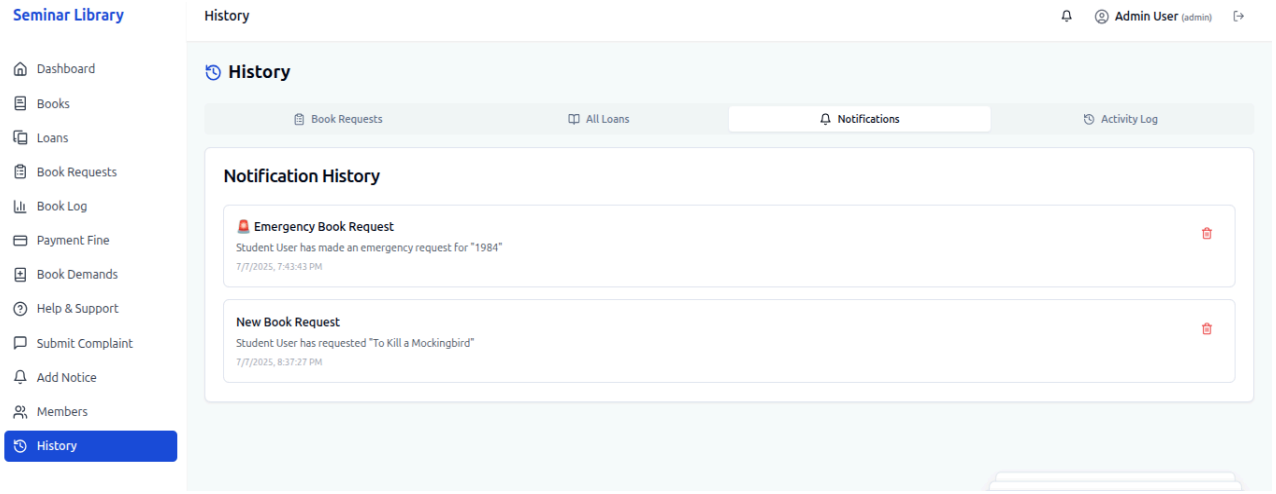

Figure 4.29: Notification History

C. Teacher Section

i) Teacher Registration and Login Page:

Teachers have their own distinct registration and login page. Teacher has to complete a registration form, providing essential information. Once registered, Teachers can seamlessly access their dedicated sections using their authorized username and password. If either the username or password is incorrect, access to the system is denied. The registration form includes vital information such as teacher name, teacher username, teacher email, teacher password, teacher designation, teacher ID number, teacher phone number and teacher address.

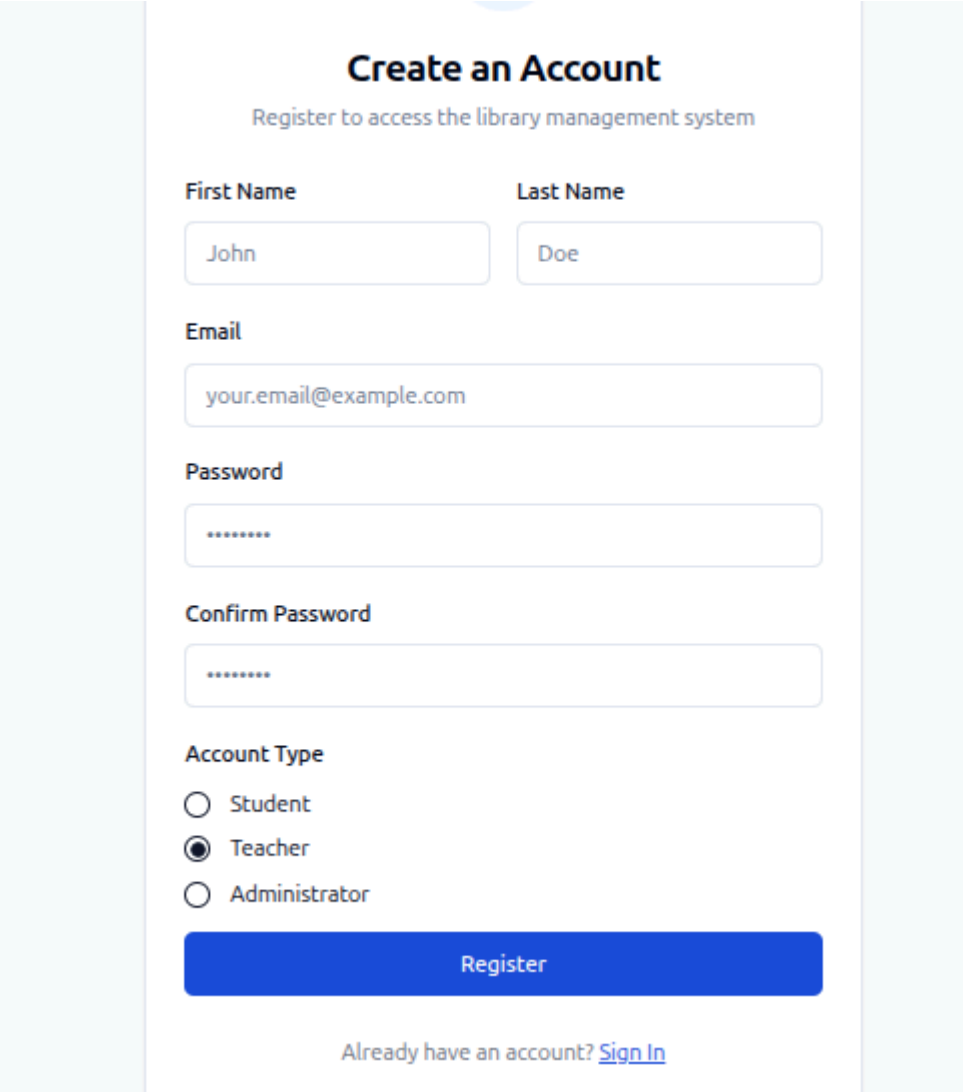

Figure 4.30: Teacher registration

ii) Teacher Dashboard

The Teacher Dashboard serves as a centralized interface designed specifically for teachers to manage and monitor library-related activities efficiently. From this dashboard, teachers can view and issue books, track borrowed items, and manage return dates. In addition to student-level access, teachers have extended privileges, such as overseeing student borrowing records, submitting or reviewing complaints, and generating reports related to library usage. The dashboard provides a clear and organized view of relevant data, empowering teachers to support academic activities while ensuring proper usage of library resources.

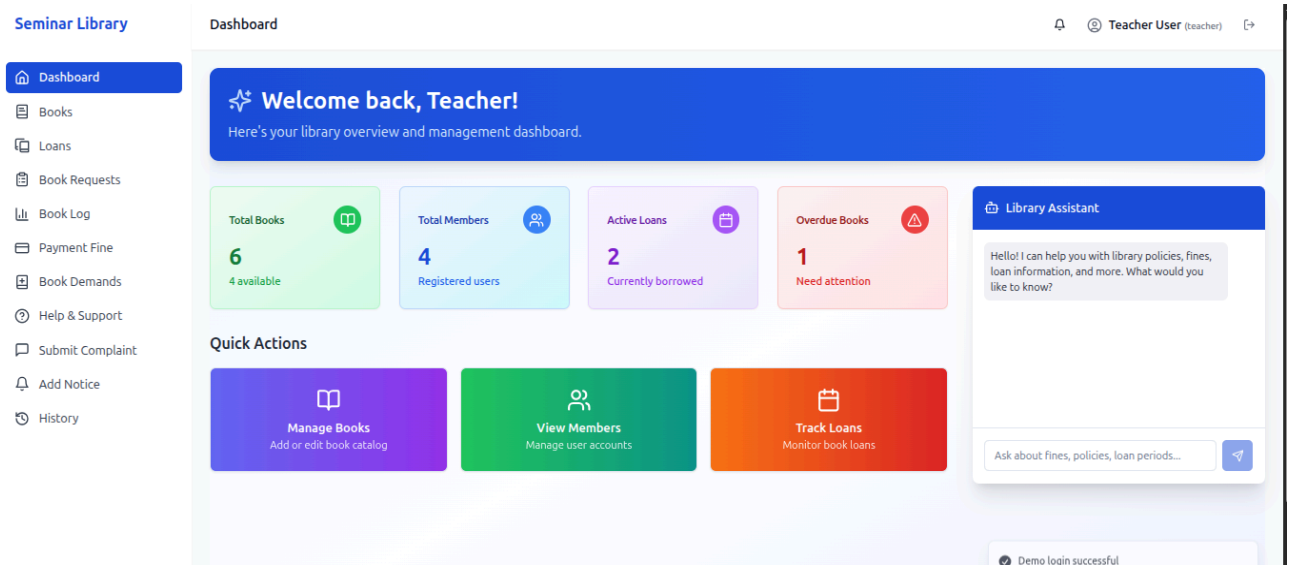

Figure 4.31: Teacher Dashboard

A teacher has access to all the functionalities available to a student, along with additional privileges specific to their role. This ensures that teachers can not only experience the platform from a student's perspective but also perform all student-level actions seamlessly. Such access supports better guidance, troubleshooting, and interaction with the learning system.

Table 4.1: Comparison Between Existing System and Developed System

Existing System	Developed system
There is a high chance of issuing a book to a non-authorized user.	Admin can easily authenticate the system user.
All the transactions of books are done manually.	All the transactions of books are done manually.
It is a time-consuming and more costly process.	Increase the efficiency and reduce the cost.
Physical inspection is required to confirm the book's availability in the library.	Easy search for books in the online system. Users need not go to the library physically to look at any book.
Maintaining user details is complex in a manual system.	Maintaining user details is complex in a manual system.

CHAPTER 5

SYSTEM TESTING AND MAINTENANCE

5.1 Test Plan

The primary goal of testing is to identify errors. Testing is the process of trying to find every potential error or fault in a product we've made. It's comparable to making sure software performs as intended and doesn't malfunction negatively. It breaks down the test items, features, tasks, and allocated duties and notes any possible hazards that call for backup plans.

It also offers a full summary of the testing process, including who will conduct the tests, what will be tested, the estimated period of testing, and the expected quality requirements. The test planning phase culminates in the creation of a high-level test plan document, which outlines the software items to be tested, the degree of tester independence, the testing environment, test case design, measurement techniques, and the rationale behind their selection. [21]

5.2 Test Activities

The objective of this system testing process is to identify all defects within the project. The program is given to a series of test inputs, and various observations are recorded to determine if the program behaves as expected or not. This Project went through two levels of testing:

I. Unit testing
II. Integration testing

5.3 Unit Testing

When a module has been developed and evaluated successfully, unit testing is performed. To test a single module, we need to offer a complete environment, which means that in addition we need:

I. The procedures belonging to other modules that the module under test calls
II. Non local data structures that module accesses
III. A procedure to call the functions of the module under test with appropriate

parameters

Test for admin

I. **Testing for admin login form:** Admin uses this form to login into his section. Admin gives the username and password, if one of those is wrong, it shows an alert message. Admin can only login into the system, if the username and password is correct.
II. **Student account addition:** In this section, admin adds new members to the library. Admin fills the student's details which are given by the student. If the same student ID or email is given, it shows an alert message. Only a unique student ID and email is acceptable.
III. **Book Addition:** In this section, admin adds new books to the library. If the same book ID is given, it shows an alert message. Only books with a unique ID are acceptable.

Test for Student login module

IV. **Test for Student login Form:** This form is used only for login of students. The student put their username and password into the login form. If those are valid, then they can login to the system otherwise there will be shown an error message.

Test for Teacher login module

V. **Test for Teacher login Form:** This form is used only for login of teacher. The teacher put their username and password into the login form. If those are valid, then they can login to the system otherwise there will be shown an error message.

5.4 Integration Testing

Integration testing involves testing various integrations of the project module by providing input in this type of testing. The primary objective is to examine the module interfaces to ensure seamless communication when one module calls another.

I. Ensures proper interaction between software modules as expected by the testing team upon integration.
II. Identifies interface errors, thereby enhancing the quality and performance of the product.
III. Validates the collaborative functionality of different software modules.
IV. Verifies functionality, performance, and reliability between integrated modules.
V. Addresses issues related to insufficient exception handling.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The application's goal is to completely digitize the Information and Communication Engineering (ICE) department's seminar library in order to get rid of the inefficiencies associated with the manual book management process. All book-related operations, including issuing, returning, and record-keeping, are carried out by hand using physical registers in the current system. This frequently leads to lost books, human error, data loss, and a significant amount of time lost for both students and library employees. This antiquated system can be replaced with the suggested digital solution, which is far more effective, accessible, and well-organized. Users can engage with the system from any location within the university network thanks to the platform's web-based interface. Using a variety of filters, such as book title, author, subject, or publication year, students can quickly find books. They can check their borrowing history, request books, and view availability in real time. This feature removes the need to physically visit the library just to check whether a book is available, saving time and improving convenience for students.The system provides a centralized dashboard that allows librarians to better manage student records, book inventories, and borrowing transactions. Through the interface, new books can be added to the system with the necessary information, categorized suitably, and made searchable. It only takes a few clicks to digitize transactions like lending and receiving books. In order to help lower the quantity of past-due returns, the system automatically keeps track of due dates and can notify students when books are about to expire. Additionally, it enables the creation of thorough reports on student usage data, book availability, and borrowing patterns, all of which can help with resource management and future library planning. Because of the application's role-based access design, librarians and students will only be able to access the features that are pertinent to them. Appropriate validation, authentication, and data backup techniques preserve security and data accuracy. Future improvements can be made to the system without interfering with the current workflow because it is scalable and lightweight.

This project improves the overall academic experience for faculty and students as well as the effectiveness of daily operations by digitizing the seminar library. It makes educational resources easily accessible, cuts down on paperwork, improves transparency in book

transactions, and minimizes delays. Over time, the system should enhance resource use, expedite library operations, and help create a more technologically sophisticated learning environment in the department.

6.2 Evaluation of Projects Strengths and Weaknesses

Strength:

I. The system responds quickly to user actions, enhancing user experience.
II. The user interface is designed to be user-friendly, ensuring ease of use.
III. Validation has been done to avoid some functional error during using the system.
IV. Search features are included across modules to allow users to easily filter data.
V. The system can verify if a book is reserved or not, improving management efficiency.
VI. Notification services remember the students about the return date of the book.

Weakness:

I. Smart card technology, which could enhance security and transactions, has not been implemented.

6.3 Suggestion for Future Enhancement

The project has a lot of promise for the future, including chances to create a comprehensive experience for the university. Smart card technology has the potential to improve transactions and security. The existing system can also be turned into an Android app, even though this project is a web-based application program to handle library details.

Lastly, journal articles and thesis/project papers from past students can be added to this system. It will help the current students learn more and complete new assignments during the research period.

REFERENCES

[1] Wikipedia Contributors, *Library Overview*, Wikipedia. Available: https://en.wikipedia.org/wiki/Library (accessed Sep. 16, 2023).

[2] C. Liu and S. Ma, "Design of Library Management System," *OALib*, vol. 5, no. 12, pp. 1–8, 2018. DOI: 10.4236/oalib.1104974.

[3] S. Ganeshan, "Library Management System," *Library Management System*. DOI: 10.37896/JXAT12.11/29777.

[4] T. Araya, T. Weldu, A. Ph, and A. Mengsteab, "Designing Web-based Library Management System," *International Journal of Engineering Research*, 2020. DOI: 10.17577/IJERTV9IS100131.

[5] A. Tripathi, "Online Library Management System," *IOSR Journal of Engineering*, vol. 2, no. 2, pp. 180–186, Feb. 2012. DOI: 10.9790/3021-0202180186.

[6] P. Kumar, R. Kumar, R. Singh, and V. P. Singh, *Library Management System Mini Project Report*, 2014.

[7] S. Ragavan, "Implementation of Automated Library Management System in the School of Chemistry Bharathidasan University using Koha Open Source Software," *International Journal of Applied Engineering Research*, vol. 1, 2010.

[8] N. Sivakumar and P. Sivaraman, "International Library Management Systems," *International Journal of Library and Information Science Research and Development*, vol. 1, no. 1, 2012.

[9] A. Mauren and O. Blessing, *Design of an Automated Library Management System for State Universities of Nigeria*, 2011.

[10] Bhupendra, S. Panwar, and V. Vaishnav, "Online Library Management System," 2011.

[11] Tutorialspoint, *SDLC Waterfall Model*, 2019. Available: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm (accessed Sep. 16, 2023).

[12] Tutorialspoint, *SDLC Iterative Model*, 2019. Available: https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm (accessed Sep. 16, 2023).

[13] Javatpoint, *Understanding the Incremental Model in Software Engineering*, 2011. Available: https://www.javatpoint.com/software-engineering-incremental-model (accessed Sep. 16, 2023).

[14] S. Djordjević-Kajan, "Fundamentals of Database Systems," *Microelectronics Journal*, vol. 28, no. 5, pp. 603–604, Jun. 1997. DOI: 10.1016/s0026-2692(97)80960-3.

[15] MDN Web Docs, *HTML: HyperText Markup Language*, Dec. 10, 2018. Available: https://developer.mozilla.org/en-US/docs/Web/HTML (accessed Sep. 16, 2023).

[16] W3C, *Overview of Cascading Style Sheets (CSS)*, 2019. Available: https://www.w3.org/Style/CSS/-Overview.en.html (accessed Sep. 16, 2023).

[17] Mozilla, *CSS: Cascading Style Sheets Documentation*, Jun. 26, 2019. Available: https://developer.mozilla.org/en-US/docs/Web/CSS (accessed Sep. 16, 2023).

[18] Wikipedia Contributors, *Bootstrap: Front-End Framework Overview*, Wikipedia, Apr. 8, 2019. Available: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) (accessed Sep. 16, 2023).

[19] H.Reactor, “Applications of JavaScript,” *Medium*, Nov. 1, 2018. Available: https://medium.com/@hackreactor/what-is-javascript-used-for-d438f3e9ca39 (accessed Sep. 15, 2023).

[20] Telerik Blogs, *Applications and Benefits of React*, Feb. 18, 2021. Available: https://www.telerik.com/-blogs/what-is-react-used-for (accessed Sep. 16, 2023).

[21] Oracle, *The Java™ Tutorials: Learn the Basics of Java Programming*, Oracle, 2020. Available: https://docs.oracle.com/javase/tutorial/ (accessed Sep. 16, 2023).

[22] Baeldung, *Introduction to Spring Boot*, Baeldung, Jan. 15, 2021. Available: https://www.baeldung.com/spring-boot (accessed Sep. 16, 2023).

[23] Pivotal, *Spring Boot Reference Documentation*, Spring.io, Apr. 8, 2019. Available: https://spring.io/projects/spring-boot (accessed Sep. 16, 2023).

[24] GeeksforGeeks, *Spring Framework Basics: A Guide to Spring Boot*, GeeksforGeeks, 2018. Available: https://www.geeksforgeeks.org/spring-framework-basics/ (accessed Sep. 16, 2023

APPENDIX

Below, I have provided some of my code to develop the project.

Priority Table (BookLog.tsx):

Code:

```
import React from 'react';
import { useQuery } from '@tanstack/react-query';
import { getBooks, getBookRequests, getLoans } from '../services/apiService';
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from
'@/components/ui/table';
import { Badge } from '@/components/ui/badge';
import { BookOpen, AlertTriangle } from 'lucide-react';

interface BookAnalytics {
  bookId: number;
  bookTitle: string;
  totalRequests: number;
  emergencyRequests: number;
  averageHoldingDuration: number;
  currentlyBorrowed: boolean;
  priority: number;
}

const BookLog: React.FC = () => {
  const { data: books = [], isLoading: booksLoading, error: booksError } = useQuery({
    queryKey: ['books'],
    queryFn: getBooks
  });

  const { data: bookRequests = [], isLoading: requestsLoading, error: requestsError } =
useQuery({
    queryKey: ['bookRequests'],
    queryFn: getBookRequests
  });

  const { data: loans = [], isLoading: loansLoading, error: loansError } = useQuery({
    queryKey: ['loans'],
    queryFn: getLoans
  });

  const isLoading = booksLoading || requestsLoading || loansLoading;
  const hasError = booksError || requestsError || loansError;

  const bookAnalytics: BookAnalytics[] = React.useMemo(() => {
    if (!books.length) return [];

    return books.map(book => {
      const bookRequestsForThisBook = bookRequests.filter(req => req.bookId === book.id);
```

```
      const loansForThisBook = loans.filter(loan => loan.bookId === book.id);

      const totalRequests = bookRequestsForThisBook.length;
      const emergencyRequests = bookRequestsForThisBook.filter(req =>
req.isEmergency).length;

      // Calculate average holding duration
      const completedLoans = loansForThisBook.filter(loan => loan.returnDate);
      const totalDuration = completedLoans.reduce((sum, loan) => {
        const checkoutDate = new Date(loan.checkoutDate);
        const returnDate = new Date(loan.returnDate!);
        const duration = Math.ceil((returnDate.getTime() - checkoutDate.getTime()) / (1000 * 60 *
60 * 24));
        return sum + duration;
      }, 0);

      const averageHoldingDuration = completedLoans.length > 0 ? Math.round(totalDuration /
completedLoans.length) : 0;

      const currentlyBorrowed = loansForThisBook.some(loan =>
        loan.status === 'BORROWED' || loan.status === 'OVERDUE'
      );

      // Priority calculation: emergency requests weight more heavily
      const priority = (totalRequests * 1) + (emergencyRequests * 3);

      return {
        bookId: book.id,
        bookTitle: book.title,
        totalRequests,
        emergencyRequests,
        averageHoldingDuration,
        currentlyBorrowed,
        priority
      };
    }).sort((a, b) => b.priority - a.priority);
  }, [books, bookRequests, loans]);

  if (isLoading) {
    return (
      <div className="flex items-center justify-center min-h-[400px]">
        <div className="text-center">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-library-blue
mx-auto mb-4"></div>
          <p className="text-gray-600">Loading book analytics...</p>
        </div>
      </div>
    );
  }

  if (hasError) {
    return (
      <div className="flex items-center justify-center min-h-[400px]">
        <div className="text-center">
```

```
        <BookOpen className="mx-auto h-12 w-12 text-red-500 mb-4" />
        <h3 className="text-lg font-medium text-red-600 mb-2">Error Loading Data</h3>
        <p className="text-sm text-gray-600">
          Unable to load book analytics. Please try refreshing the page.
        </p>
      </div>
    </div>
  );
}

 return (
  <div className="space-y-6">
    <div className="border rounded-md">
      <Table>
        <TableHeader>
          <TableRow>
            <TableHead>Book Title</TableHead>
            <TableHead>Total Requests</TableHead>
            <TableHead>Emergency Requests</TableHead>
            <TableHead>Avg. Duration (Days)</TableHead>
            <TableHead>Status</TableHead>
            <TableHead>Priority Score</TableHead>
          </TableRow>
        </TableHeader>
        <TableBody>
          {bookAnalytics.length === 0 ? (
            <TableRow>
              <TableCell colSpan={6} className="text-center py-12">
                <BookOpen className="mx-auto h-12 w-12 text-muted-foreground" />
                <h3 className="mt-4 text-lg font-medium">No book data available</h3>
                <p className="text-sm text-muted-foreground">
                  Analytics will appear here once books and requests are available
                </p>
              </TableCell>
            </TableRow>
          ) : (
            bookAnalytics.map((book) => (
              <TableRow key={book.bookId}>
                <TableCell className="font-medium">
                  <div className="flex items-center gap-2">
                    <BookOpen className="h-4 w-4 text-library-brown-light" />
                    <span className="truncate max-w-[200px]" title={book.bookTitle}>
                      {book.bookTitle}
                    </span>
                  </div>
                </TableCell>
                <TableCell>
                  <span className="px-2 py-1 rounded text-xs font-medium bg-blue-100
text-blue-800">
                    {book.totalRequests}
                  </span>
                </TableCell>
                <TableCell>
                  {book.emergencyRequests > 0 ? (
```

```
                  <div className="flex items-center gap-1">
                    <AlertTriangle className="h-3 w-3 text-red-500" />
                    <span className="px-2 py-1 rounded text-xs font-medium bg-red-100
text-red-800">
                      {book.emergencyRequests}
                    </span>
                  </div>
                ) : (
                  <span className="text-gray-400 text-sm">0</span>
                )}
              </TableCell>
              <TableCell>
                {book.averageHoldingDuration > 0 ? (
                  <span className="text-gray-700">{book.averageHoldingDuration}</span>
                ) : (
                  <span className="text-gray-400 text-sm">N/A</span>
                )}
              </TableCell>
              <TableCell>
                <Badge
                  variant={book.currentlyBorrowed ? "default" : "outline"}
                  className={book.currentlyBorrowed ? "bg-orange-100 text-orange-800
border-orange-200" : ""}
                >
                  {book.currentlyBorrowed ? "Borrowed" : "Available"}
                </Badge>
              </TableCell>
              <TableCell>
                <Badge
                  variant={book.priority >= 10 ? "destructive" : book.priority >= 5 ? "default" :
"secondary"}
                  className="text-xs"
                >
                  {book.priority}
                </Badge>
              </TableCell>
            </TableRow>
          ))
        )}
      </TableBody>
    </Table>
  </div>
 </div>
 );
};

export default BookLog;
```

Chatbot code:

```
import React, { useState } from 'react';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Send, Bot } from 'lucide-react';
import { FINE_RATE_PER_DAY } from '../services/fineService';

interface ChatMessageProps {
  message: {
    id: string;
    text: string;
    isUser: boolean;
  };
}

const ChatMessage: React.FC<ChatMessageProps> = ({ message }) => {
  return (
    <div className={`flex ${message.isUser ? 'justify-end' : 'justify-start'}`}>
      <div
        className={`rounded-lg p-2 text-sm max-w-xs ${
          message.isUser
            ? 'bg-library-blue text-white'
            : 'bg-gray-100 text-gray-800'
        }`}
      >
        {message.text}
      </div>
    </div>
  );
};

const ChatBot = () => {
  const [messages, setMessages] = useState([
    {
      id: '1',
      text: 'Hello! I can help you with library policies, fines, loan information, and more. What would you
like to know?',
      isUser: false,
    },
  ]);
  const [inputMessage, setInputMessage] = useState('');
  const [isTyping, setIsTyping] = useState(false);

  const getLibraryResponse = (userMessage: string): string => {
    const lowerMessage = userMessage.toLowerCase();

    // Fine-related queries
    if (lowerMessage.includes('fine') || lowerMessage.includes('overdue') ||
```

```
lowerMessage.includes('penalty')) {
      return `📚 **Overdue Fines Policy:**\n\n• Fine rate: ${FINE_RATE_PER_DAY} BDT per day for overdue books\n• Fines start accumulating the day after the due date\n• You can check your current fines in the "My Books" or "Loans" section\n• **SMS Notifications:** We'll send you text messages 3 days before due date and daily after overdue\n• Please return books on time to avoid fines!`;
    }

    // Notification queries
    if (lowerMessage.includes('notification') || lowerMessage.includes('sms') || lowerMessage.includes('text') || lowerMessage.includes('reminder')) {
      return `📱 **SMS Notification System:**\n\n• **3 Days Before Due:** "Please return your book before overdue for 3 days"\n• **Daily After Overdue:** "Please return your book, your fine is X BDT"\n• Notifications sent to your registered phone number\n• Make sure your phone number is updated in your profile\n• Fine rate: ${FINE_RATE_PER_DAY} BDT per day after due date`;
    }

    // Loan period queries
    if (lowerMessage.includes('loan') || lowerMessage.includes('borrow') || lowerMessage.includes('due date') || lowerMessage.includes('return')) {
      return `📖 **Loan Policies:**\n\n• Standard loan period: 14 days\n• Books can be renewed if no one else has requested them\n• You can view your current loans and due dates in "My Books"\n• **SMS reminders:** 3 days before due date and daily after overdue\n• Please return books on time to avoid fines of ${FINE_RATE_PER_DAY} BDT per day`;
    }

    // Book request queries
    if (lowerMessage.includes('request') || lowerMessage.includes('reserve') || lowerMessage.includes('emergency')) {
      return `📝 **Book Request Policy:**\n\n• You can request available books through "Book Requests"\n• Emergency requests can be made for urgent academic needs\n• Requests are reviewed by library staff\n• You'll be notified when your request is approved or rejected`;
    }

    // Membership queries
    if (lowerMessage.includes('member') || lowerMessage.includes('registration') || lowerMessage.includes('account')) {
      return `👤 **Membership Information:**\n\n• Students and teachers can register for library membership\n• Admin approval is required for new accounts\n• Members can borrow books, make requests, and access digital resources\n• Keep your contact information updated for notifications`;
    }

    // Library hours and general info
    if (lowerMessage.includes('hours') || lowerMessage.includes('open') || lowerMessage.includes('time') || lowerMessage.includes('schedule')) {
      return `🕒 **Library Hours:**\n\n• Monday-Friday: 8:00 AM - 8:00 PM\n• Saturday: 9:00 AM - 5:00 PM\n• Sunday: Closed\n• Extended hours during exam periods\n• Holiday schedules may vary`;
    }

    // Contact and help
    if (lowerMessage.includes('contact') || lowerMessage.includes('help') || lowerMessage.includes('support')) {
      return `📞 **Contact Information:**\n\n• Library Desk: +880-XXX-XXXX\n• Email: library@seminar.edu.bd\n• Location: Ground Floor, Main Building\n• For technical issues, contact IT support\n• Staff available during library hours`;
```

```
    }

    // General library policies
    if (lowerMessage.includes('policy') || lowerMessage.includes('rule') ||
lowerMessage.includes('regulation')) {
      return `📋 **Library Policies:**\n\n• Maintain silence in reading areas\n• Handle books with care\n•
No food or drinks near books\n• Mobile phones on silent mode\n• Return books by due date\n• Report
damaged books immediately\n• Fine: ${FINE_RATE_PER_DAY} BDT per day for overdue books`;
    }

    // Default response
    return `I can help you with:\n\n• Overdue fines (${FINE_RATE_PER_DAY} BDT/day)\n• SMS
notifications (3 days before due & daily after overdue)\n• Loan periods and renewals\n• Book requests and
reservations\n• Library hours and contact info\n• Membership information\n• General library
policies\n\nWhat specific information do you need?`;
  };

  const handleSendMessage = async (e: React.FormEvent) => {
    e.preventDefault();
    if (!inputMessage.trim()) return;

    const userMessage = {
      id: Date.now().toString(),
      text: inputMessage,
      isUser: true,
    };

    setMessages((prevMessages) => [...prevMessages, userMessage]);
    setInputMessage('');
    setIsTyping(true);

    // Simulate typing delay
    await new Promise((resolve) => setTimeout(resolve, 1000));

    const botResponse = {
      id: Date.now().toString() + '-bot',
      text: getLibraryResponse(inputMessage),
      isUser: false,
    };

    setMessages((prevMessages) => [...prevMessages, botResponse]);
    setIsTyping(false);
  };

  return (
    <div className="bg-white rounded-lg shadow-xl w-full h-96 flex flex-col">
      {/* Chat Header */}
      <div className="bg-library-blue text-white p-4 rounded-t-lg flex items-center gap-2">
        <Bot className="h-5 w-5" />
        <span className="font-medium">Library Assistant</span>
      </div>

      {/* Messages Area */}
      <div className="flex-1 p-4 overflow-y-auto space-y-3">
```

```
      {messages.map((message) => (
        <ChatMessage key={message.id} message={message} />
      ))}
      {isTyping && (
        <div className="flex items-center gap-2 text-gray-500">
          <Bot className="h-4 w-4" />
          <span className="text-sm">Assistant is typing...</span>
        </div>
      )}
    </div>

    {/* Input Area */}
    <div className="p-4 border-t">
      <form onSubmit={handleSendMessage} className="flex gap-2">
        <Input
          value={inputMessage}
          onChange={(e) => setInputMessage(e.target.value)}
          placeholder="Ask about fines, policies, loan periods..."
          className="flex-1"
          disabled={isTyping}
        />
        <Button
          type="submit"
          size="icon"
          disabled={isTyping || !inputMessage.trim()}
          className="bg-library-blue hover:bg-library-blue-dark"
        >
          <Send className="h-4 w-4" />
        </Button>
      </form>
    </div>
  </div>
 );
};

export default ChatBot;
```