VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY
THE INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

ONLINE LIBRARY MANAGEMENT SYSTEM

By
Thai Gia Lac - ITITIU19152
Dr. Huynh Kha Tu

A thesis submitted to the School of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Computer Science

Ho Chi Minh City, Vietnam
2023

ONLINE LIBRARY MANAGEMENT SYSTEM

APPROVED BY SUPERVISOR

_____

Huynh Kha Tu, Ph.D.

APPROVED BY COMMITEE

_____

_____

_____

_____

_____

THESIS COMMITTEE

ACKNOWLEGMENTS

I would like to express my gratitude to Dr. Huynh Kha Tu for her guidance and unwavering support during the completion of my thesis. The success and quality of this study owe a deal to Dr. Tu's expertise, insightful feedback, and dedicated mentorship.

Without her knowledge, the field comments, and constructive criticism my thesis wouldn't have reached its level of refinement or improvement. Driven by her commitment to excellence and eagerness to share her knowledge I have experienced growth both academically and personally.

I am truly indebted to Dr. Tu for being a supervisor and mentor who has shaped my perspective on the topic while encouraging me on a journey of learning. Without her guidance completing my thesis would have been a task, which's why I am forever grateful.

Finally, Dr. Huynh Kha Tu, I extend my appreciation for your wisdom dedication in assisting me in overcoming obstacles and unwavering encouragement, throughout my academic endeavors.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The Online Library Management System provides a complete platform for students to explore a vast database of books, search for titles, and get full information about each publication. The system incorporates user-friendly functionalities, guaranteeing a seamless and intuitive navigation experience. Within the virtual environment of the website, students may easily access information about book availability, make reservations, and keep track of their borrowing history.

The adoption of this Online Library Management System is set to transform the conventional library paradigm, providing students with a simple and effective method to oversee their reading materials. With digital technologies, educational institutions may optimize the book-borrowing procedure, leading to an enhanced academic experience for students.

CHAPTER 1

INTRODUCTION

1.1. Background.

The conventional library concept is experiencing a revolutionary transition in the fast-expanding world of education and technology. Considering the internet and digital platforms, educational institutions must urgently adopt creative solutions that improve the accessibility and effectiveness of library services. The proposed Online Library Management System for student book borrowing is a solution to this need, with the goal of transforming the way students engage with and make use of library resources. The widespread availability of internet access and the common use of digital devices by students make it an ideal setting for the installation of an Online Library Management System. This method utilizes the ease and accessibility of the internet, enabling students to conveniently browse, search, and borrow books. The transition from a tangible to a digital library not only broadens the accessibility of educational materials but also corresponds to the modern learning preferences of technologically proficient pupils. In addition, the suggested system acknowledges the significance of data precision and timely information updates. Through the consolidation of book management in an online setting, institutions may establish a centralized database that accurately represents the real-time status of each book. This empowers students to make well-informed decisions on their borrowing options. By including user verification, the system guarantees the security of student accounts and their borrowing history, therefore promoting a dependable and credible system.

1.2. Problem Statement

Despite the progress in technology and the growing use of digital instructional materials, several academic institutions still struggle with inefficiencies in their library administration systems. Conventional book borrowing techniques often rely on manual procedures, have restricted availability, and lack up-to-date information, which obstructs students' smooth access to crucial educational resources. Physical libraries are subject to limitations about their operation hours and physical capacity. Students may have difficulties in obtaining library resources outside the specified operating hours, which might hinder their capacity to borrow essential books for their academic endeavors. Conventional library systems sometimes depend on human tracking techniques, resulting in mistakes in the availability status of volumes. This may lead to

exasperation for students who may go on needless visits to the library, only to discover that a desired book is unavailable.

The inherent immutability of conventional libraries may result in restricted involvement among students. An Online Library Management System may improve user engagement by offering an interactive platform for browsing, searching, and finding a wider array of educational materials. Safeguarding student accounts and personal information is of utmost importance in the era of digital technology. Conventional library systems cannot have strong user authentication mechanisms, which might lead to issues over the privacy and security of user data.

By implementing an Online Library Management System specifically designed for student book borrowing, we can address these difficulties and create a library environment that is more accessible, efficient, and safe. Using technology, educational institutions may improve the entire academic experience for students by aligning library services with modern learning preferences and propelling the institution into the digital era.

1.3. Scope and Objectives

Scope:

The proposed Online Library Management System has a broad scope, with the goal of transforming the existing library paradigm and enhancing the student experience in accessing educational materials. The system will include:

- User-Friendly Interface: The creation of an interface that is easy to use and understand, accessible via web browsers, to ensure that students with different levels of technical skills may navigate it easily.
- Reservation and Borrowing Features: This system simplifies the operations of reserving and borrowing books, enabling students to effectively handle their borrowing activities, access their borrowing history, and get prompt information about due dates and book returns.
- The Extensive Book Catalog is a comprehensive and searchable database that offers a wide range of educational resources, such as books, journals, and multimedia materials. It is designed to meet the different requirements and interests of the student population.

- Responsive Design ensures compatibility with a range of devices, such as laptops, tablets, and smartphones, to guarantee accessibility across many platforms and foster a versatile learning environment.

Objectives:

- Improve Accessibility: Enable unrestricted access to educational materials at any time and from any location, therefore minimizing limitations associated with physical library space and working hours.
- Promote User Engagement: Develop an interactive platform that actively involves students in the study of instructional resources, fostering a proactive and immersive learning experience.
- Encourage Flexibility: Create a flexible design that adjusts to different devices, accommodating the varied technological preferences of students and fostering a smooth user experience.
- Enhance Security: Employ strong user authentication protocols to protect student accounts and personal data, fostering trust in the online library system's security.
- Enable interaction: Guarantee a seamless interaction with current library management systems and academic databases, eliminating interruptions and enabling a seamless shift to the online platform.

1.4. Assumption and Solution

In order for the Online Library Management System to function effectively, it is essential that both students and the academic institution have access to a dependable and fast internet connection. Furthermore, it assumes that students own digital devices such as PCs, laptops, tablets, or smartphones, which allow them to interact with the online library platform effortlessly. Moreover, it is presumed that the educational institution has the requisite infrastructure to safely maintain and sustain the online system. My studies and research at the International University have given me a reasonable understanding of how to construct and create an online library management system. Furthermore, a measurement of the processes' correctness in practice cannot be provided by this research at this time. With any luck, I'll be able to have another chance to make the software more practically useful in the near future.

1.5. Structure of thesis

The structure of my thesis organized as follows:

- Chapter 1 provides a background of the chosen topic, as well as my approach to the problem, proposed solution direction, scope, and goal, along with some underlying assumptions.
- Chapter 2 covers the basic theories regarding the research topic from journal, book, and the internet.
- Chapter 3 describes the implementation of this application in system design.
- Chapter 4 shows how tools and techniques are applied to build the application.
- Chapter 5 presents and evaluates the results of the thesis, discussing its strengths and weaknesses.
- Chapter 6 summarizes what has been accomplished in this thesis and proposes future improvements.

CHAPTER 2

LITURATURE REVIEW/RELATED WORK

2.1. Librarything.com

2.1.1. General introduction:

LibraryThing is a social cataloging web application that allows individuals to catalog their personal libraries and connect with other readers. LibraryThing has evolved into a thriving community of bibliophiles, and its functionalities appeal to both occasional readers and passionate book enthusiasts. It remains a widely used site for those seeking to oversee and share their reading encounters with others.

2.1.2 Feature analysis:

Feature	Evaluate
Search by book categories	By simply inputting book titles, authors, or ISBNs, users may effortlessly create a database of their books. Afterwards, LibraryThing compiles and presents facts on every book, such as cover art, publishing information, and reviews left by readers.
Online Book Reservation	libraryThing allows users to add books they want to borrow to the cart and checkout to schedule user information, time to borrow and return books.

Ratings and Reviews	Users can evaluate through comments and vote the number of stars of the book to review and improve the quality of services and books in the library.
Cart Management	LibraryThing allows users to manage their book selections in the cart, add or remove books, and modify the borrowing details such as the pick-up date and duration. The cart will display the total number of books, due dates, and applicable late fees.
Borrowing History	LibraryThing allows user access to their borrowing history, which includes a list of books they have borrowed in the past, along with the dates of borrowing and return.
Personal Profile Management	LibraryThing allows users to update their personal information, such as Member name, About My Library, Real Name and Location. They may also have the option to provide feedback or suggestions regarding the online library management system.

Table 1: Librarything features

2.2. Bookshare.org

2.2.1 General introduction

To help those who have trouble reading print find books and other reading resources, Bookshare has created an online digital library. Bookshare provides a wide variety of electronic books in accessible forms including big type, braille, and audiobooks. It was founded by the charitable group Benetech. People with learning difficulties, visual impairments, and other print challenges will be able to read more easily on this platform. Everyone, regardless of age, may find something to read in Bookshare's vast collection of fiction, nonfiction, instructional materials, and more.

2.2.2 Feature analysis:

Pros	Cons
○ Offers alternative forms such as audiobooks and braille, promoting inclusion for persons who have difficulties reading printed materials. ○ Provides a wide-ranging and comprehensive assortment of e-books, including educational resources, works of fiction, and non-fiction literature. ○ Promotes active participation within the community, enabling users to exchange and endorse books in order to enrich the reading experience. ○ This platform is especially advantageous for students who have difficulties reading printed materials, since it provides access to textbooks and educational resources. ○ Users may conveniently and flexibly access books on several devices.	○ Only those with certain impairments are allowed, which restricts the number of users. ○ Although there are free memberships, there are also paid subscriptions for persons who do not qualify, which may restrict access for some users. ○ The website's UI might be intimidating for some visitors owing to the plethora of features and settings available. ○ Availability of titles in different accessible formats may be limited, which might restrict alternatives for particular users. ○ The presence of copyright limitations may affect the availability of some titles, hence influencing the entire range of material that may be accessed.

Table 2: Bookshare features

2.3. IDE in project

2.3.1 Visual Studio Code

Figure 1: Visual Studio Code

One free and open-source code editor that Microsoft created is Visual Studio Code, or VS Code for short. It has rapidly risen to prominence as a top choice for developers looking for a code editor. Visual Studio Code's strong features improve the coding experience, and the program is recognized for being lightweight and speedy.

- Collaborate and code remotely: Collaborate remotely with your lecturers or students by using the free Live Share plugin. Modify and troubleshoot your code in real-time and use the chat and phone functionalities to inquire about or exchange ideas collaboratively. If you are engaged in a collaborative project or conducting a teaching session, you can extend invitations to many individuals to participate in your session and collaborate on coding tasks.
- Fix errors as you code: While coding, Visual Studio Code provides recommendations to automatically finish lines of code and offers fast remedies for common errors. Additionally, the debugger in VS Code may be used to sequentially execute each line of code and gain comprehension of its functionality. Explore tutorials on using the debugger for Python, Java, and JavaScript/TypeScript/Node.js programming languages.

- Compare changes in your code: Utilize the integrated source control feature to save your work throughout time, so preventing any loss of progress. Observe a visual representation that allows for direct comparison of different versions of your code at various time periods.

2.3.2 ReacJs

2.3.2.1 Introduction ReactJS

Jordan Walke of Facebook developed the open-source JavaScript toolkit ReactJS to construct user interfaces for mobile and online applications. 2011 saw the debut of React on Facebook's newsfeed. It was utilized on Instagram in 2012 and released to the public in 2013. Applications that alter their data often and dynamically are best suited for React. Without re-rendering the whole page, it may display the UI elements that are changing. This results in a substantially quicker user experience. Let's go over the introduction to ReactJS and its associated elements in case you're still unclear about what it is all about.

Figure 2: ReactJS

2.3.2.2 Advantages of using ReactJS

With Reacts component-based architecture, developers can design modular and reusable components. This modular strategy encourages component reuse across many application components and improves code structure and maintainability. React uses a Virtual DOM to

streamline the rendering process. React determines the most effective method to update the real DOM when changes are made to a virtual version of the DOM. Performance is enhanced and fewer adjustments are required, therefore. Because react enforces a unidirectional data flow, it's simpler to comprehend how changes to the data impact the state of the application. This method makes debugging easier and helps avoid frequent data synchronization problems.

2.3.3 ViteJs

2.3.3 1 ViteJs

Vite.js is a modern frontend development tool created by Evan You, offering a fast and efficient environment for web application development. It is framework-agnostic, allowing developers to use various frontend frameworks or plain HTML, CSS, and JavaScript projects. Vite.js supports modern technologies like TypeScript, CSS pre-processors like Sass and Less, and even bundling for production using Rollup, making it suitable for various project types.

Vite.js provides a paradigm change in the way we approach frontend development build tools. Its focus on speed, adaptability, and a contemporary development experience makes it an appealing alternative for developers wanting to optimize their processes and build high-performance online apps.

Figure 3: ViteJS

2.3.3 1 Advantages of using ViteJs

ViteJS is noted for its exceptionally fast development server. It employs ES modules to serve code in development without bundling, leading in speedy start times and quicker feedback loops during development.

In production, ViteJS improves the build process by utilizing Rollup for bundling. It provides highly optimized and minified code for fast loading and increased website performance.

ViteJS is a comprehensive plugin system that enables developers to enhance and tweak its capabilities. This makes it simple to interface with multiple tools, preprocessors, and other technologies according to project needs.

ViteJS has acquired appeal among the developer community, resulting to an active community that produces plugins, tools, and support. The project also maintains thorough documentation, making it simpler for developers to get started and discover answers to common difficulties.

2.3.4 CSS framework

Bootstrap, Tailwind CSS, and Material UI are three prominent technologies used for online UI development. All three give a big collection of pre-designed components, helping to decrease development effort and speed up the user interface creation process. Bootstrap and Tailwind CSS are CSS frameworks, offering classes and components to let you effortlessly design attractive and easy-to-maintain web interfaces.

2.3.4.1 Bootstrap reactjs

Using Bootstrap with React entails importing Bootstrap's CSS styles and JavaScript components into the React project, allowing developers to exploit Bootstrap's responsive grid system and UI elements while benefitting from React's component-based design. This combination simplifies the development process, increases code reusability, and helps the building of contemporary and visually attractive user interfaces.

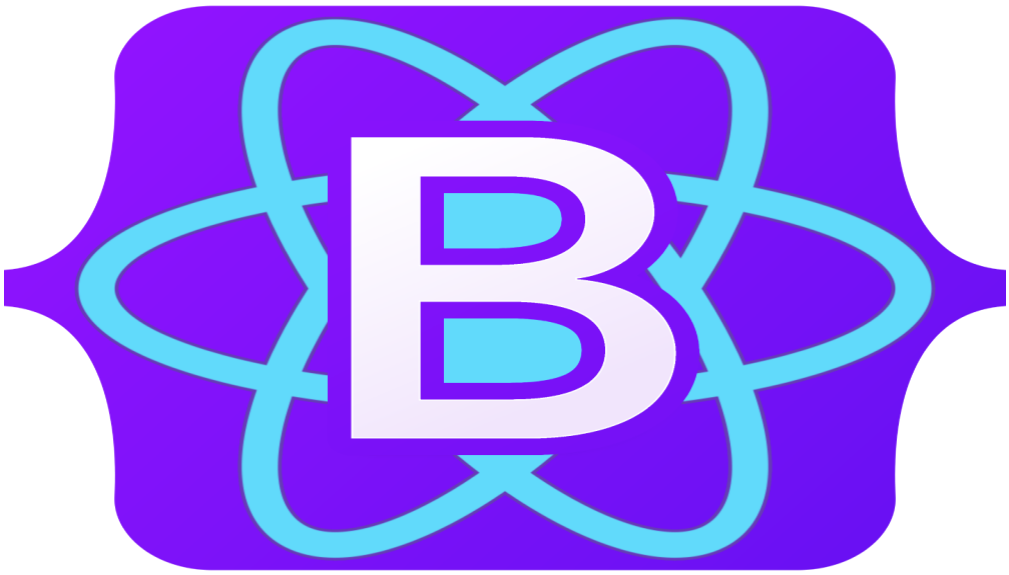


Figure 4: React-Bootstrap

2.3.4.2 TailwindCSS

Tailwind CSS is a low-level, utility-first CSS framework that streamlines the building of contemporary, highly flexible user interfaces. It includes a collection of utility classes that may be added straight to HTML markup, allowing for speedy creation and simple modification. Tailwind is extremely flexible, enabling developers to tailor the framework to include exactly the styles they require, decreasing file size and maximizing speed. It also offers responsive design features, enabling layouts to adapt to multiple screen sizes.

Tailwind's utility classes are clear and quick to understand, making it suited for a broad variety of tasks. Developers may install Tailwind CSS using npm or yarn, customize it according to their requirements, and apply its utility classes directly in HTML or inside a CSS file using build tools like PostCSS.

Figure 5: Tailwindcss

2.3.4.3 Material UI

Material-UI is a React UI framework that follows Google's Material Design standards, giving pre-designed and customisable components for contemporary online apps. It follows to Google's values, delivering a sophisticated visual language for user interfaces. Material-UI's modular React components enable for simple integration into projects. They may be personalized and adjusted to fit the application's appearance, including colors and font.

Material-UI's components are responsive, delivering a consistent user experience across multiple devices and screen sizes. It also promotes accessibility, making it accessible to a broad variety of users, including those with impairments. The framework's vast community offers resources, tutorials, and community assistance. Developers may install Material-UI with npm or yarn, and the library is continuously updated to line with the newest Material Design requirements.

Figure 6: Material UI

2.3.5 NodeJS

2.3.5.1 Introduction NodeJS

Node is an open-source, cross-platform runtime environment that allows developers to construct server-side tools and apps in JavaScript. It is meant for usage outside of a browser environment and supports more standard OS APIs like HTTP and file system libraries. Node delivers exceptional speed, minimizing time spent on context shift across languages. JavaScript, a relatively new programming language, provides advances in language design compared to older established web-server languages. Other popular languages can also compile/convert into JavaScript.

The node package manager (npm) gives access to hundreds of thousands of reusable packages and can automate development toolchains. Node is portable and available on several platforms, including Microsoft Windows, macOS, Linux, Solaris, FreeBSD, OpenBSD, WebOS, and NonStop OS. It features a significant third-party ecosystem and development community. Node may be used to construct a basic web server using the Node HTTP module.

Figure 7: Node JS

2.3.5.2 Advantages of using NodeJS

Node.js is a popular choice for server-side development due to its fast execution, asynchronous and non-blocking I/O model, single language for both client and server, extensive NPM ecosystem, community support, scalable capabilities, cross-platform compatibility, real-time capabilities, ease of learning, and active development. It is based on the V8 JavaScript engine from Google Chrome, which converts JavaScript code straight into machine code, resulting in greater speed. Node.js has an event-driven, non-blocking I/O approach, enabling it to handle several concurrent connections concurrently without waiting for one action to complete before initiating another. Its single language enables for increased code reuse and a uniform development experience throughout the whole application stack.

The platform is cross-platform, running on many operating systems, making it appropriate for a broad variety of deployment situations. Its event-driven design and support for WebSockets allow the construction of responsive and interactive features. JavaScript is a commonly utilized language for many developers, making it simple to migrate to server-side programming. Node.js is open-source and actively maintained by a devoted community, ensuring it remains current with evolving trends in web development.

2.3.6 Express JS

2.3.6.1 Introduction ExpressJS

ExpressJS is a fast, simple web framework for Node.js technology, aimed to ease the development process of server-side applications using JavaScript. It is created on top of Node.js, enabling flexibility and speed, making it simpler for developers to construct scalable and responsive online apps. ExpressJS enables developers to route, manage, and process HTTP requests with simplicity using middleware. Its unopinionated nature means there is no structured method to write code, enabling developers to construct single-page, multi-page, and hybrid web apps. Its straightforward API supports developers in constructing online applications, and it includes feature-rich capabilities including routing, template engines, and middleware support. Overall, ExpressJS is a good alternative for developers aiming to construct quick, scalable, and dependable online apps with minimum scripting.

Figure 8: ExpressJS

2.3.6.2 Advantaages of using ExpressJS

ExpressJS is a fast, unopinionated, and simple web framework for Node.js technology that streamlines the development process of server-side applications utilizing JavaScript as its programming language. It is created on top of Node.js, enabling flexibility and speed, making it simpler for developers to construct scalable and responsive online apps. ExpressJS enables developers to route, manage, and process HTTP requests with simplicity, and provides a simple and straightforward API that supports developers in constructing online apps.

Express has various features, including simplicity, middleware architecture, powerful routing system, flexibility, template engine support, active community and ecosystem, built-in

HTTP utility functions, robust error handling mechanisms, performance, and broad acceptance. The framework's simplicity enables developers to arrange their applications according to their tastes, while its middleware design allows for simple integration of other modules and libraries.

Express also supports several template engines, including as EJS, Pug, and Handlebars, enabling developers to select the one that best meets their requirements and tastes. Its vibrant community offers resources, tutorials, and third-party middleware, making it easier for developers to discover answers to common issues and remain informed with best practices.

2.3.7 MongoDB

2.3.7.1 Introduction MongoDB

MongoDB is a document-oriented NoSQL database used for big volume data storage. Instead of employing tables and rows as in the classic relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the primary unit of data in MongoDB. Collections include collections of documents and function which is the equivalent of relational database tables. MongoDB is a database which came into light in the mid-2000s.

Figure 9: MongoDB

2.3.7.2 Advantages of using MongoDB

- Each database comprises collections which in turn contain documents. Each document might be distinct with a changing number of fields. The size and content of each document might be different from each other.
- The document format is more in line with how developers design their classes and objects in their different programming languages. Developers will frequently argue that their classes are not rows and columns but have a clear structure with key-value pairs.
- The rows (or documents as called in MongoDB) doesn't need to have a schema established beforehand. Instead, the fields may be built on the fly.
- The data architecture accessible inside MongoDB enables you to describe hierarchical connections, to store arrays, and other more complicated structures more simply.
- Scalability – The MongoDB environments are extremely scalable. Companies throughout the globe have established clusters with some of them operating 100+ nodes with about millions of records inside the database.

CHAPTER 3

METHODOLOGY

3.1 Overview

The methods that went into creating a Process Management System are detailed in this section. In order to determine how to approach and implement the system's operations and features, user needs are examined. An overview of the system's design, including use case diagrams, class diagrams, sequence diagrams, and database design, is provided in the System Design section.

3.2 User requirement analysis

The objective of this project is to develop an online library management system that caters specifically to students and requires a dedicated system management department. Hence, the intended users are limited to two roles: student and administrator.

- Student: Students can register, log in, access the book library, borrow books, manage book carts, and examine their borrowing history. Modify profile to enable reception of alerts. Additionally, you have the option to engage in a conversation with the administrator.

- Administrator: Administrators has the authority to oversee the whole of the book library, including all details pertaining to each student member, and can modify the status of books. Additionally, you have the ability to oversee both book borrowing requests and account registration requests for new students. Access information and engage in conversations with other students.

3.3 Use case diagram.

An online library management system's use case diagram depicts the many methods via which users engage with the system and the distinct features it offers. It offers a comprehensive summary of the system's behavior and aids stakeholders in comprehending the system's needs from a user's standpoint. Below is a concise elucidation of the objective and prospective applications for an internet-based repository of books and resources.

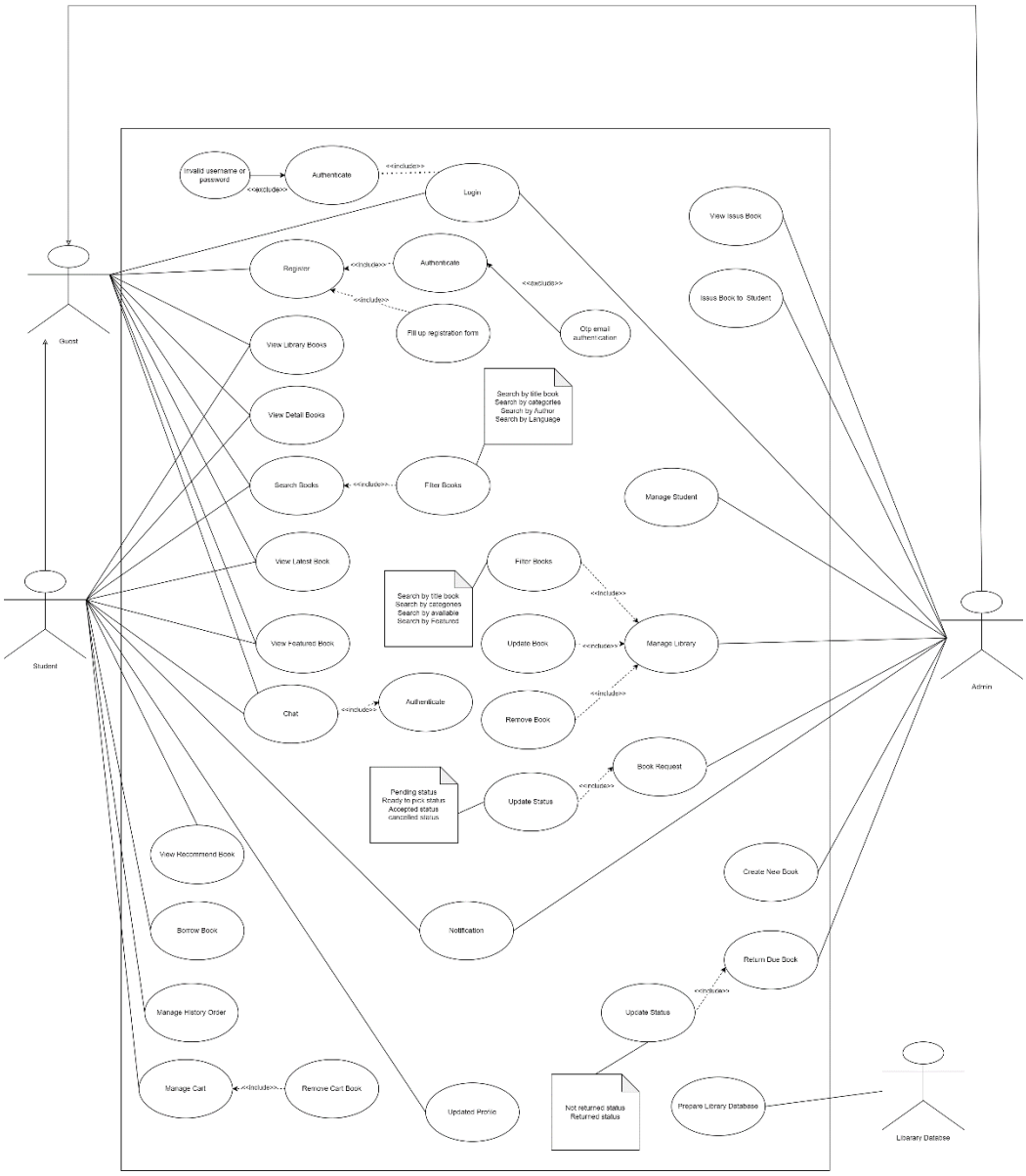


Figure 10: Use case diagram.

3.4 Use case diagram and descriptions.

USE CASE 1

Name: Log in.

Identifier: UC1.

Actor: Guest

Inputs: Email and password.

Outputs: User login successfully

Basic course:

Normal Flow	1. The login form is shown by the system. 2. User enters email and password. 3. User presses a login icon. 4. System transmits login credentials to the system.. 5. The system authenticates information, displays a successful login message. 6. Library Website redirect to home screen.
Alternative Flow	
Exception Flow	3.a.1 In the event that the user does not complete all fields, the system shows an error message.. 5.a.1 The system displays the message "Invalid email or password" if you enter the wrong account or password.

Table 3: UC1: Log in

Precondition: A valid email account and password were supplied by the user.

Post condition: Tokens are given to the user so they may authenticate and utilize the system's services.

USE CASE 2

Name: Sign up account.

Identifier: UC2.

Actor: Guest

Inputs: User information.

Outputs: Account already register success.

User story: I would want to make an account as a guest.

Basic course:

Normal Flow	1. System shows login button in home page 2. User choose the signup button 3. System displays the form to fill in the account information 4. User fills in the required fields 5. The user clicks on the accept Terms and Conditions box 6. System sends registration information to the system

	7. The system sends OTP authentication information to the user's email 8. System display the OTP input screen 9. User enters OTP and press "Confirm" button 10. System shows successful account creation message and redirect to login page
Alternative Flow	9.a.1 User selects Resend OTP 9.a.2 System resend OTP and show notification that the code has been sent to email
Exception Flow	5.a.1 If any fields are left blank, the system shows an error message. 5.b.1 System displays error message if same account email 8.a.1 System returns to the login screen 9.a.1 System shows wrong OTP message if wrong code is entered

Table 4: UC2: Sign up account.

Precondition: A user is not logged in to the system

Post condition: None.

USE CASE 3

Name: Forgot password**.**

Identifier: UC3.

Actor: Student

Inputs: Email and phone number.

Outputs: Password has been recovered successfully.

Trigger: The user pressed forgot password button below the form on the login page.

User story: As a user, I want to retrieve my password when I forgot my account password.

Basic course:

Normal Flow	1.The system displays forgot password button in login page 2. User choose the forgot password button 3.The system displays forgot password form 4. User input the email and phone

	5.The system gets information and redirect to recover password page 6. User fill the password and confirm password 7. The system redirects to the login page and shows a message that the password reset was successful.
Alternative Flow	
Exception Flow	5.a.1 System displays error message if the password and confirm password is not match

Table 5: UC3: Forgot password.

Precondition: The user forgot the password.

Post condition: Password change.

USE CASE 4

Name: Search for books.

Identifier: UC4.

Actor: Guest

Inputs: Book's title.

Outputs: The input title appears in a list of similar books for the user to peruse.

User story: To locate the book I want to borrow, I as a user want to be able to search for it by typing in its title.

Precondition: None.

Post condition: The system lists books according to the data that users have filtered.

Basic course:

Normal Flow	1. system finds popular books and the latest books. 2. The system displays the book on the screen. 3. The system waits for users to enter keywords of title in the search bar. 4. When the user enters a keyword, the system will filter for a few seconds, and the application sends a notification to search for related books. 5. After receiving the outcome, the system shows it on the screen.

Alternative Flow	4.a.1 Even without typing a term, the user continues to click Search. 4.b.1 User inputs a book that the computer is unable to locate. 4.b.2 The system will display “No Results”
Exception Flow	

Table 6: UC4: Search for books.

USE CASE 5

Name: View book detail.

Identifier: UC5.

Actor: Guest

Inputs: Users have the option to choose a particular book from the list in order to see its individual information.

Outputs: The user can view all detailed information of the book such as title, author, language, availability, and description.

User story: As a user, I want to see brief information about a book and a short description of the book before deciding to borrow it.

Precondition: None.

Post condition: The system displays all information about the book users have chosen to view.

Basic course:

Normal Flow	1. The user clicks the View button on each book on the library page. 2. The website displays detailed information of the book such as title, author, language, availability, and description.
Alternative Flow	1a. On the home page and library page, each book has a View button for users to see book information
Exception Flow	

Table 7: UC5: View book detail.

USE CASE 6

Name: View latest book.

Identifier: UC6.

Actor: Guest

Inputs: None.

Outputs: users can view the latest updated books on the home page.

User story: As a user, I want to see the latest books updated on the home page so they can be easily found.

Precondition: None.

Post condition: Users can see the latest books on the home page.

Basic course:

Normal Flow	1. The library system will retrieve data from the most recently created books to display on the home page. 2. The system will list the newest books in the "Latest Books" section.
Alternative Flow	
Exception Flow	

Table 8: UC6: View latest books.

USE CASE 7

Name: View featured books.

Identifier: UC7.

Actor: Guest

Inputs: None.

Outputs: Users can view the most featured books on the home page.

User story: As a user, I want the most featured books to be displayed on the home page to have the best choices.

Precondition: None.

Post condition: Students can view the most featured books.

Basic course:

Normal Flow	1. The system retrieves data from books with outstanding sections. 2. The system will display the most prominent books in the "Featured Books" section.
Alternative Flow	2a. Users can filter by categories of books

Exception Flow	

Table 9: UC7: View featured books.

USE CASE 8

Name: View recommended books.

Identifier: UC8.

Actor: Student

Inputs: None.

Outputs: Students can see recommended books on the home page.

User story: As a student, I want to see recommended books that are similar to the type of books I often borrow.

Precondition: Guest must be logged in.

Post condition: Students can view recommended books.

Basic course:

Normal Flow	1. Guest logs in to an account with the role of student. 2. The system retrieves student information about borrowed books and recommends books of similar genres. 3. The system will list recommended books in the "Recommended Books" section on the home page.
Alternative Flow	
Exception Flow	

Table 10: UC8: View recommended books.

USE CASE 9

Name: View popular books

Identifier: UC9.

Actor: Guest

Inputs: None

Outputs: users can see popular books on the library page.

User story: As a user, I want to see the books that most people borrow and list them so it's easy to choose the best ones.

Precondition: There must be at least one student borrowing a book to display popular books.

Post condition: Users can view the most popular books.

Basic course:

Normal Flow	1. The system retrieves data about the most borrowed books. 2. The system will list popular books in the "Popular Books" section on the library page.
Alternative Flow	
Exception Flow	

Table 11: UC9: View popular books.

USE CASE 10

Name: View documents

Identifier: UC10.

Actor: Guest

Inputs: None

Outputs: Users can view all learning materials on the documents page and download those files to their personal devices.

Precondition: None.

Post condition: Users can view documents and download them.

Basic course:

Normal Flow	1. Users click on the "Documents" button in the sidebar to be navigated to the document viewing page. 2. The system will list all learning documents along with the document names.
Alternative Flow	2.a Users can click the view button next to the document to preview it. 2.b Users can download that file in .pdf format.
Exception Flow	

Table 12: UC10: View documents.

USE CASE 11

Name: Bot Chat

Identifier: UC11.

Actor: Guest

Inputs: Users choose items and questions according to the guest's role with bot chat

Outputs: Users will be answered by the chat bot based on the answer sets and questions for the guest role.

Precondition: None.

Post condition: Users have their questions answered and instructions to solve problems on the library website.

Basic course:

Normal Flow	1. Guest clicks on the message icon on every page on the library website 2. The chat bot board will be displayed for guests to ask questions based on the system's question data set. 3. The chat bot system will respond immediately to the guest
Alternative Flow	2.a There are multiple options for guests to choose from based on the guest role's chat bot question set. 3.a The chat bot's response set is also based on the guest role's data.
Exception Flow	

Table 13: UC11: Bot Chat (Guest).

USE CASE 12

Name: Bot Chat

Identifier: UC12.

Actor: Student

Inputs: Student choose items and questions according to the student's role with bot chat

Outputs: Student will be answered by the chat bot based on the answer sets and questions for the student role.

Precondition: Guest must be logged in.

Post condition: Student have their questions answered and instructions to solve problems on the library website.

Basic course:

Normal Flow	1. Users log into the system with the student role. 2. Student clicks on the message icon on every page on the library website 3. The chat bot board will be displayed for students to ask questions based on the system's question data set. 4. The chat bot system will respond immediately to the student
Alternative Flow	3.a There are multiple options for students to choose from based on the students role's chat bot question set. 4.a The chat bot's response set is also based on the student role's data.
Exception Flow	

Table 14: UC12: Bot Chat (Student) .

USE CASE 13

Name: Create request book.

Identifier: UC13.

Actor: Student

Inputs: None.

Outputs: The student has successfully requested to borrow books.

Precondition: Users must be logged in.

Post condition: Created a request to borrow books.

Basic course:

Normal Flow	1. Students click the "Reset" button on each book to create a request 2. The system retrieves student data, book information and creates a request to borrow books
Alternative Flow	

Exception Flow	2.a.1 If you create a request for borrowed books, you will be notified "Book already Requested"

Table 15: UC13: Create request book.

USE CASE 14

Name: View history.

Identifier: UC14.

Actor: Student

Inputs: None.

Outputs: The student's entire book borrowing history includes information and status of the book.

Precondition: Guest must be logged in.

Post condition: Students can see the entire process of borrowing their books on the history page. library website.

Basic course:

Normal Flow	1. Students who click on the profile button on the navbar will be navigated to the book borrowing history page 2. The system will retrieve student data and list all books and the status of borrowed books
Alternative Flow	2.a Students can view the book's processing status as accepted, canceled, or ready for pickup 2.b Students can see the date the book was borrowed, the due date for returning the book, and the status of the book being returned to the school
Exception Flow	

Table 16: UC14: View history.

USE CASE 15

Name: Cancel request history.

Identifier: UC15.

Actor: Student

Inputs: None.

Outputs: The book request item was successfully canceled in the history page.

Precondition: Students must create a request to borrow books.

Post condition: The requested book is canceled.

Basic course:

Normal Flow	1. Students go to the history page 2. The system will retrieve student data and display all book borrowing history information 3. If the book loan request has not been processed, there will be a "Cancel" button next to it for students to cancel the book loan request. 4. Book requests are cancelled
Alternative Flow	3.a Books with a status other than "PENDING" will not have a "Cancel" button.
Exception Flow	

Table 17: UC15: Cancel request history.

USE CASE 16

Name: Remove request history.

Identifier: UC16.

Actor: Student

Inputs: None.

Outputs: Successfully removed history entries that have completed book returns.

Precondition: The student's book return status is "TRUE".

Post condition: Remove returned books history.

Basic course:

Normal Flow	1. Students go to the history page 2. The system will retrieve student data and display all book borrowing history information 3. Students press the "REMOVE" button for history items whose book purchase history data is "TRUE" 4. Book requests are removed
Alternative Flow	3.a Book borrowing status is "FALSE" and there will be no "REMOVE" button for students

Exception Flow	

Table 18: UC16: Remove request history.

USE CASE 17

Name: View detail profile.

Identifier: UC17.

Actor: Student

Inputs: None.

Outputs: Displays all student details such as username, phone, email, and total Book.

Precondition: Students must log in to the system.

Post condition: Show all student account information.

Basic course:

Normal Flow	1. Students click on the profile icon on the library system's navbar. 2. Students continue to click on the "My Details" button, the system will directly navigate to the profile page. 3. On the profile page, the system displays all student account information
Alternative Flow	
Exception Flow	

Table 19: UC17: View detail profile.

USE CASE 18

Name: Update profile.

Identifier: UC18.

Actor: Student

Inputs: New information such as name, phone, and password

Outputs: New information such as name, phone and password have been successfully updated.

Precondition: Students must log in to the system.

Post condition: Displays information in student profiles that have been successfully updated.

Basic course:

Normal Flow	1. Students click on the profile icon on the library system's navbar. 2. Students continue to click on the "My Details" button, the system will directly navigate to the profile page. 3. On each right side of the username, phone and password information, there is an "Edit" button for students to click on. 4. Then, for each information the student wants to change, the system will display the corresponding modal 5. Students enter new information to make changes and press the "Save changes" button. 6. Student profile information updated successfully
Alternative Flow	5.a In the password modal, students must re-enter the old password and "confirm" the new password.
Exception Flow	5.b .1 If a student enters the old password incorrectly, the system will make an error 5.b.2 If the student enters the newly learned password correctly according to the validated password form, a system error will be displayed 5.b.3 If the student enters a confirm password that does not match the new password, the system will report an error

Table 20: UC18: Update profile.

USE CASE 19

Name: Manage books

Identifier: UC19.

Actor: Admin

Inputs: None.

Outputs: List all created book lists.

Precondition: Users must log in under the "admin" role.

Post condition: Show all list of books in database.

Basic course:

Normal Flow	1. Admin click on the "Manage Books" button on the left sidebar.

	2. The system will navigate the admin to the book management page. 3. The system shows the entire list of books created in the database.
Alternative Flow	
Exception Flow	3.a If there are no books in the database, the system will display "0 Book Data"

Table 21: UC19: Manage books.

USE CASE 20

Name: Search created books.

Identifier: UC20.

Actor: Admin

Inputs: Admin can search by book title, categories, featured and available.

Outputs: List books by title keyword or by categories, featured and available.

Precondition: None.

Post condition: Displays books according to the filtered admin.

Basic course:

Normal Flow	1. Admin enters keywords in the title field, or selects options in categories, featured, availability of the books 2. Press the "Search" button on the left of the filter bar 3. The system will start retrieving filtered data 4. The system lists books according to the information the admin has filtered
Alternative Flow	2.a Admin can press the "Clear Filters" button next to the "Search" button to filter again from the beginning
Exception Flow	4.a If the book filter admin has no data, the system will return "0 Book result's"

Table 22: UC20: Search created books.

USE CASE 21

Name: Edit book.

Identifier: UC21.

Actor: Admin

Inputs: New contents of the book such as title, Category, Author, Available, Language and Description.

Outputs: The book's content was "updated successfully".

Precondition: Admin must log in to the system.

Post condition: Changed book information is updated.

Basic course:

Normal Flow	1. Admin presses the "View Details" button on the book management page to be navigated directly to the book update page 2. Admin fills in the information that needs to be updated for the book 3. Click the "Update" button below to update book information 4. The book notification system has been updated successfully
Alternative Flow	
Exception Flow	

Table 23: UC21: Edit books.

USE CASE 22

Name: Edit image book.

Identifier: UC22.

Actor: Admin

Inputs: Upload new photos of the book.

Outputs: The book's photo has been successfully updated.

Precondition: Admin must log in to the system.

Post condition: The book's photo has been updated.

Basic course:

Normal Flow	1. Admin presses the "View Details" button on the book management page to be navigated directly to the book update page

	2. Admin clicks on the "choose file" button below the image that needs to be changed 3. The system will need to select photos from personal device data 4. The admin presses a few "Update image" buttons to update the book's article image 5. The system will notify you that the image has been updated successfully
Alternative Flow	
Exception Flow	3.a If the admin chooses files other than the image file format, the system will report an error

Table 24: UC22: Edit image books.

USE CASE 23

Name: Delete book.

Identifier: UC23.

Actor: Admin

Inputs: None.

Outputs: The book has been deleted successfully.

Precondition: Admin must log in to the system.

Post condition: Book data is deleted from the database.

Basic course:

Normal Flow	1. Admin presses the "View Details" button on the book management page to be navigated directly to the book update page 2. Admin select the "Delete" button below for book information 3. The system will have a modal to ensure that the admin wants to delete the book 4. admin press the "Yes" button to accept 5. The system notifies that the book data has been successfully deleted

	6. The system loads and navigates directly to the book management page to check the book data has been deleted
Alternative Flow	4.a Admin press the "No" button to cancel deleting the book
Exception Flow	

Table 25: UC23: Delete book.

USE CASE 24

Name: Add new book.

Identifier: UC24.

Actor: Admin

Inputs: Title of the book, Author, Description, Category, Language and select the "True" or "False" option of the two available fields, Featured, Upload the book's cover image.

Outputs: The book was created successfully.

Precondition: Admin must log in to the system and fill in all the information in the book.

Post condition: The book is added to the database.

Basic course:

Normal Flow	1. admin click on the "Add new book" button on the left sidebar. 2. The system will navigate the admin to the book creation page. 3. Admin fills in all fields of the book. 4. Admin clicks on the "Submit" button to create the book. 5. The system will notify you that the book has been successfully created
Alternative Flow	
Exception Flow	3.a If the admin does not fill in all the book information, the system will notify you that you need to complete the form to create the book

Table 26: UC24: Add new book.

USE CASE 25

Name: Update status request.

Identifier: UC25.

Actor: Admin

Inputs: admin selects the options "Ready to pick", "Accepted", and "Canceled"

Outputs: Request status updated successfully.

Precondition: Admin must log in to the system.

Post condition: Book request status is updated in the system.

Basic course:

Normal Flow	1. Admin click on the " Books Request's" button on the left sidebar. 2. The system will navigate the admin to the books request's page. 3. The system will list all student book processing statuses. 4. Admin presses the "Update" button, the system will create 3 options "Ready to pick", "Accepted" and "Canceled." 5. Admin chooses one of the 3 options and press update to update the book request status. 6. The system will notify you that the book request status has been successfully updated
Alternative Flow	5.a.1 All students who borrow books are in "Pending" status. 5.a.2 Admin selects the "Ready to pick" status to notify students 5.a.3 Admin selects the "Accepted" status to end the book request process 5.a4 Admin selects "Canceled" status to cancel student's book request
Exception Flow	

Table 27: UC25: Update status request.

USE CASE 26

Name: View users.

Identifier: UC26.

Actor: Admin

Inputs: None

Outputs: List of all registered students.

Precondition: Admin must log in to the system.

Post condition: Displays the entire list of registered students in the database.

Basic course:

Normal Flow	1. Admin click on the " View User's" button on the left sidebar. 2. The system will navigate the admin directly to the view users page 3. The system displays all student lists in the database
Alternative Flow	3.a If there is no student data, the system will display "0 Student signup"
Exception Flow	

Table 28: UC26: View users.

USE CASE 27

Name: View user details.

Identifier: UC27.

Actor: Admin

Inputs: None.

Outputs: Student details and book borrowing history.

Precondition: Admin must log in to the system.

Post condition: Displays all student information in the data and the student's book request history.

Basic course:

Normal Flow	1. The admin clicks the "View Details" button on the view users page for each student 2. The system will directly navigate the admin to the student detail view page

	3. The system will display all student information such as email, phone, name, total books, book borrowing history and email verification status.
Alternative Flow	3.a.1 If the student has verified their email with the OTP code, the system will display "Verified", or if they have not verified, "UnVerified" 3.a.2 If the student does not have a book transaction history, the system will display "0 Book data"
Exception Flow	

Table 29: UC27: View user details.

USE CASE 28

Name: Update email users.

Identifier: UC28.

Actor: Admin

Inputs: New email from student

Outputs: Email has been updated successfully.

Precondition: Users must authenticate the OTP code with their personal gmail.

Post condition: The student's new email has been successfully updated into the database.

Basic course:

Normal Flow	1. Admin clicks the "update" button on the user detail view page 2. The system will display a modal for the admin to enter the student's new email 3. Admin clicks on the "update" button in the model to be navigated directly to the gmail OTP authentication form 4. Admin enters the OTP code to authenticate the student's gmail 5. Admin presses the "update email" button to change the student's email 6. The notification system has successfully updated student emails 7. email is updated into library system data
Alternative Flow	3.a Admin can click to resend the OTP code

Exception Flow	2.a If the admin enters the wrong gmail form form, an error will be reported 3.b If the authentication code is incorrect, the system will notify "OTP invalid"

Table 30: UC28: Update email users.

USE CASE 29

Name: View issue books.

Identifier: UC29.

Actor: Admin

Inputs: None

Outputs: List of student book statuses that have not been returned.

Precondition: Admin must log in to the system.

Post condition: Displays a list of students who have not returned books to the library system.

Basic course:

Normal Flow	1. Admin click on the " Issued Books" button on the left sidebar. 2. The system will navigate the admin directly to the issued books page. 3. The system displays a list of students who have not returned books to the library system
Alternative Flow	
Exception Flow	

Table 31: UC29: View issue books.

USE CASE 30

Name: Create request user.

Identifier: UC30.

Actor: Admin

Inputs: book title, student email and book id

Outputs: Successfully created a request to borrow books for students.

Precondition: Admin must log in to the system, input the correct book id and student email

Post condition: Displays the request to create a book for a successful student.

Basic course:

Normal Flow	1. Admin click on the " Issue a Book to User" button on the left sidebar. 2. The system will navigate the admin directly to the issue book to user page. 3. Admin enters the name of the book the student needs to borrow 4. The system will find the id of that book 5. admin copy the book id and enter it in the "Book ID" field 6. Admin enters the email of the student who needs to borrow a book 7. Admin clicks the "Submit" button to create a book loan request for the student 8. The system notifies that the issue to user has been successfully created
Alternative Flow	
Exception Flow	7.a.1 If you enter the wrong book id, the system will report an error 7.a.2 If the student's email is entered incorrectly, the system will report a non-existent error

Table 32: UC30: Create request user.

USE CASE 31

Name: Update Due Books.

Identifier: UC31.

Actor: Admin

Inputs: Change the return status option.

Outputs: The student's payment status has been successfully updated.

Precondition: Admin must log in to the system.

Post condition: Displays the student's book return status changed.

Basic course:

Normal Flow	1. Admin click on the " Return Due Books" button on the left sidebar.

	2. The system will navigate the admin directly to the returned books page. 3. Admin selects the "Returned" option in the status update section 4. Admin press the "update" button to update the status 5. The system notifies the student that the book has been successfully updated
Alternative Flow	
Exception Flow	

Table 33: UC31: Update due books.

3.5 Database diagram

At the heart of this system lies the "User" entity, representing the individuals interacting with the library system. Users are classified based on their roles, such as students, faculty, librarians, or administrators. This categorization facilitates tailored access and permissions, ensuring a personalized experience for each user. The "Book" entity encapsulates the core information about each literary work in the library. Details like title, author, ISBN, genre, and publication year are tracked, alongside practical information like the total and available copies. This allows the system to efficiently manage the inventory and availability of books.

A crucial aspect of the system is the "Borrowing Transaction" entity, creating a link between users and books. Each transaction records the borrowing date and the anticipated return date, ensuring a transparent and traceable borrowing history. The one-to-many relationship between users and borrowing transactions reflects the dynamic nature of user-library interactions. This diagram provides an overview of the main entities and their relationships in an online library management system.

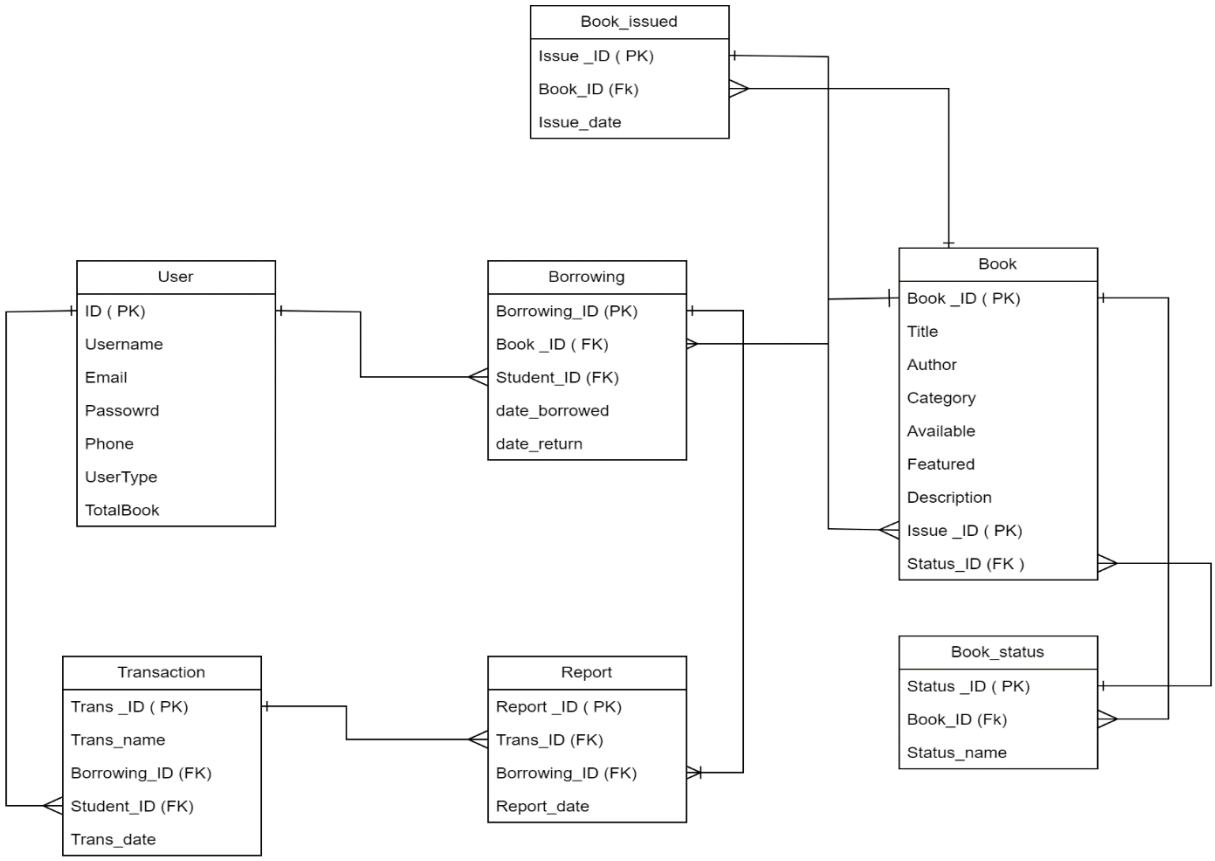


Figure 11: Database diagram

3.6 Sequence diagram

A sequence diagram is a visual depiction of the interactions between several components in an online library management system, such as the student and the library system. It enhances the lucidity of interactions, simplifies the comprehension of the sequence of events, and functions as a means of communication among stakeholders. Stakeholders may use it to pinpoint distinct elements, such as the student, library system, and external systems or databases. Sequence diagrams depict the sequential arrangement of messages and exchanges, facilitating comprehension of the temporal components of the process. Visualizing the series of interactions helps to identify possible faults or areas of failure, which assists in refining and enhancing system design.

Additionally, it offers a comprehensive explanation of the system's structure and aids in identifying the points of interaction between the user interface, application logic, and the database or other services that are involved in processing the book request. Sequence diagrams serve the purpose of verifying requirements and documenting the dynamic behavior of the system.

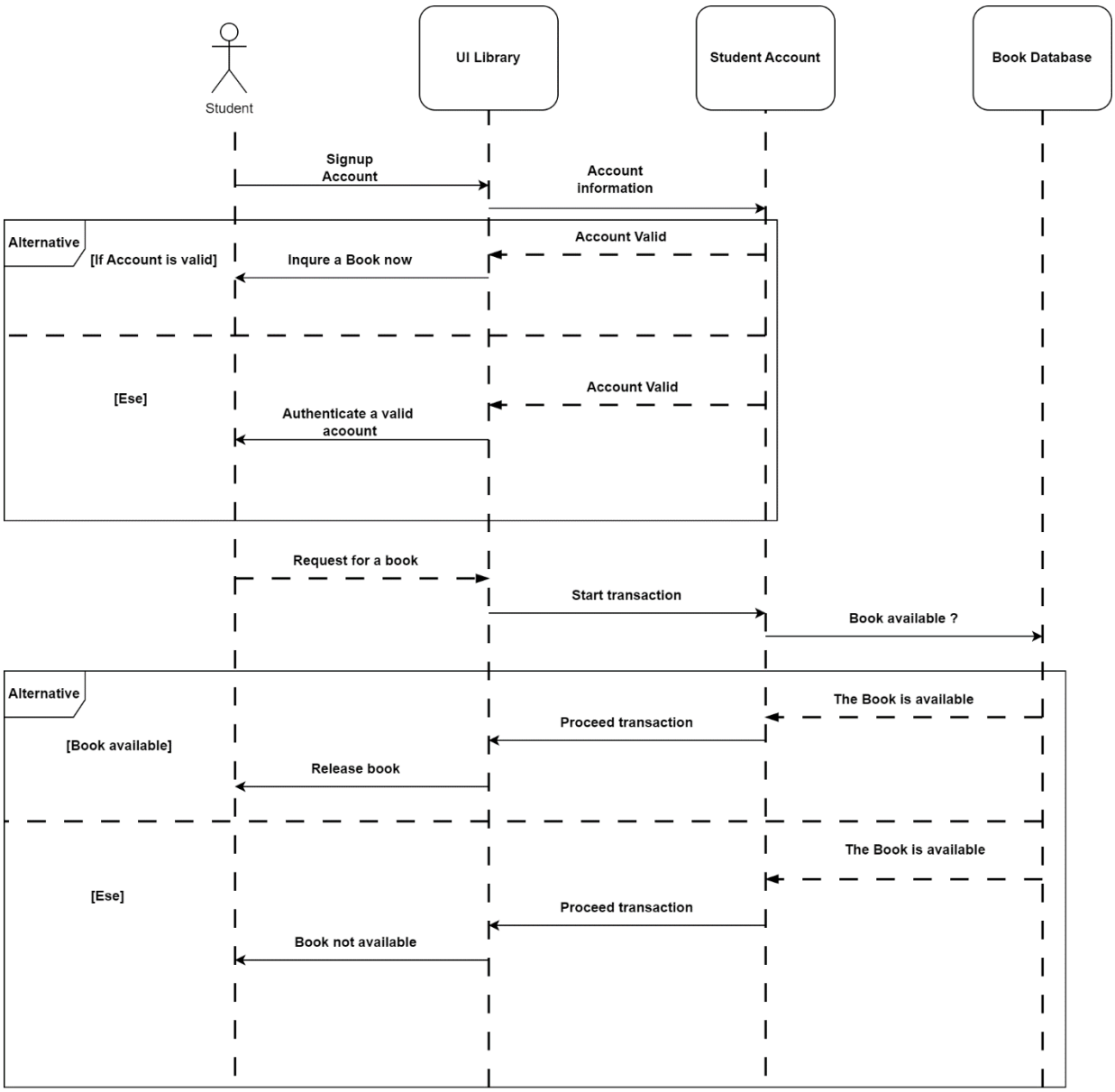


Figure 12: Sequence diagram: Request for a book

CHAPTER 4

IMPLEMENT AND RESULT

4.1 Implement

4.1.1 Front end

In the quickly expanding field of web development, selecting the correct tools and technologies is vital for producing efficient and scalable apps. This article explains how to use Vite, a cutting-edge front-end build tool, to put up a React project for an online library management system.

Prior to going through the process of configuring a React project using Vite, it is important to comprehend why Vite is used for this project. Because Vite uses ES module imports to facilitate quicker development builds, it stands out for its speed. Hot Module Replacement (HMR) is another feature of its development server that makes development easier.

- ❖ Setting up the Vite Project:

The initial step requires installing Vite globally on your system. You may use npm or yarn for this reason.

- npm install -g create-vite or yarn create vite

Create a new Vite project with the following command, using the "react" preset for our React-based application.

- create-vite my-library-management --template react.

Change into the project directory.

- cd my-library-management

Install the essential dependencies, including the React library and any additional packages needed for your project.

- npm install or yarn install.

Project Structure: Explore the project structure generated by Vite. Key files include src/main.jsx as the entry point and src/App.jsx for the main application component. Familiarize yourself with the public directory for static assets and the src directory for your application code.

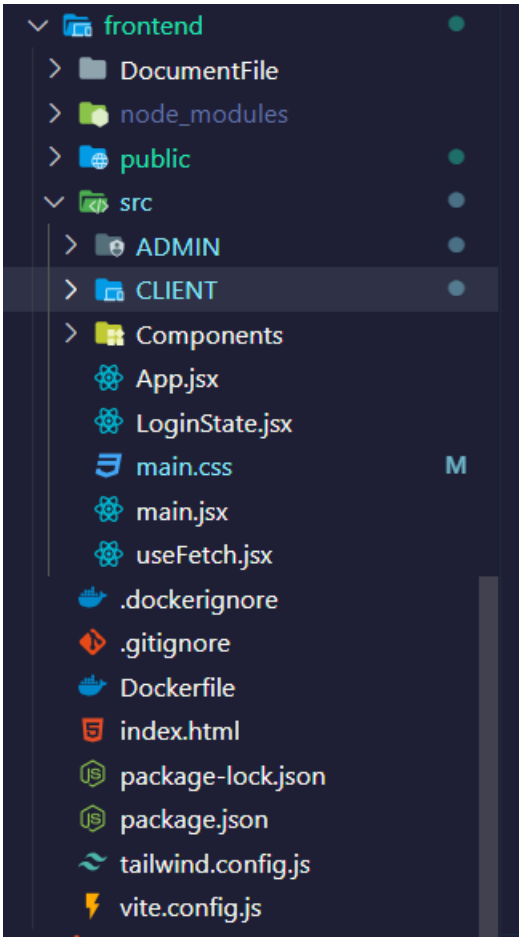


Figure 13: Front end folder structure

- **Document folder** to store all pdf files to upload to the document page, where students can view and download documents to study.
- Create a **public folder** to save all images of the online library project for easy import into other folders.
- The App.jsx file is the main UI rendering file of the entire system and decentralizes there by checking the student's logged in status and at the same time checking the userTupe to navigate to pages with similar roles.

```
  const [userType, setUserType] = useState('')

  const updateBookCharges = async () => {
    // hits api endpoints that runs book fine charge if not returned
    const response = await axios.get(UPDATE_BOOK_FINE)
    // console.log(response.data.message)
  }

  useEffect(() => {
    updateBookCharges()
    const storedUserType = localStorage.getItem('userType')
    setUserType(storedUserType)
  }, [])

  return (
    <React.Fragment>
      <Toaster />
      {userType === 'admin_user' ? <AdminAPP /> : <ClientApp />}
    </React.Fragment>
  )
}
```

Figure 14: App.jsx file

- **Folder CLIENT** will include all student and guest pages, **Folder ADMIN** will include all admin pages.

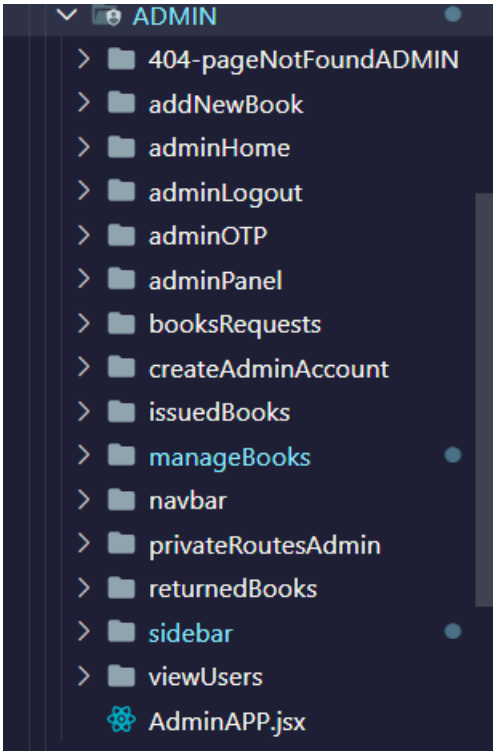


Figure 15: Admin page folder structure

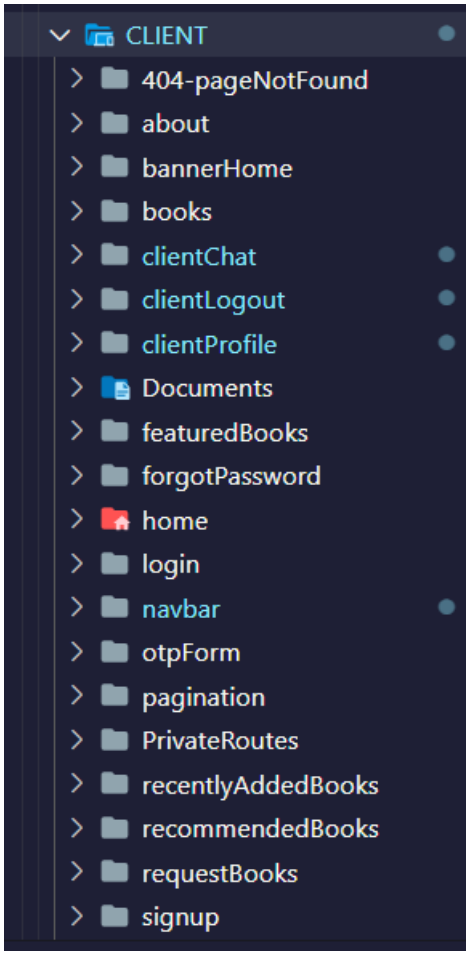


Figure 16: student and guest page folder structure

- Setup with Tailwind CSS

Tailwind CSS works by scanning all of your HTML files, JavaScript components, and any other templates for class names, producing the associated styles and then writing them to a static CSS file.

It's quick, adaptable, and dependable – with zero-runtime.

+ Installation Tailwindcss using PostCSS plugin is the most seamless way to integrate it with build tools.

+ Install tailwindcss and its peer dependencies via npm, and create your tailwind.config.js file.

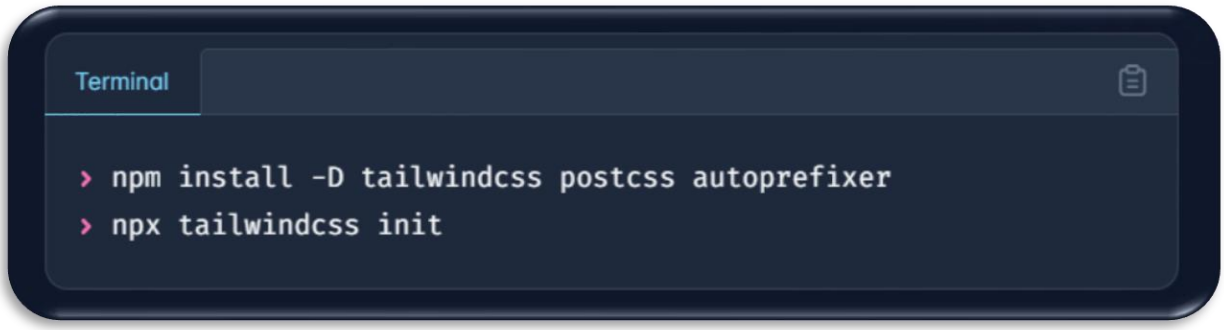


Figure 17: Install Tailwind CSS

+ Add Tailwind to your PostCSS configuration. Add tailwindcss and autoprefixer to your postcss.config.js file, or wherever PostCSS is configured in your project.

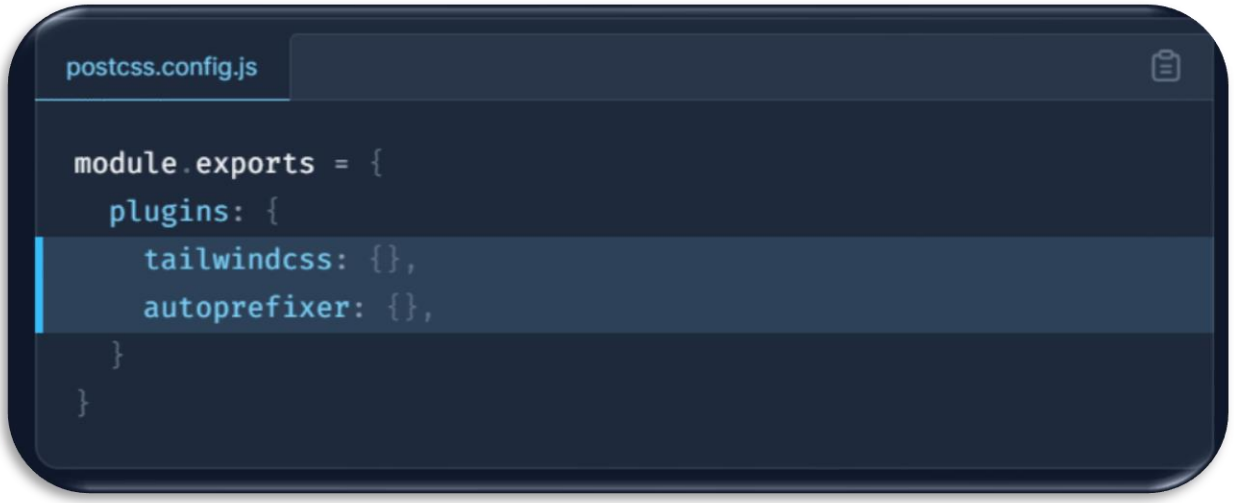


Figure 18: Add Tailwind to your PostCSS configuration

+ Configure your template paths. Add the paths to all of your template files in your tailwind.config.js file.

postcss.config.js

```
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
  }
}
```

Figure 19: Configure your template paths.

+ Add the Tailwind directives to your CSS. Add the @tailwind directives for each of Tailwind's layers to your main CSS file.

main.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Figure 20: Start your build process.

+ Start your build process. Run your build process with npm run dev or whatever command is configured in your package.json file.

Terminal

```
› npm run dev
```

Figure 21: Start your build process.

4.1.2 Back end

The selection of backend technology in contemporary web development significantly influences the effectiveness and speed of apps. When considering an online library management system, the combination of Node.js, Express.js, and MongoDB stands out as a powerful stack, providing several capabilities and benefits.

Node.js is well recognized for its ability to handle non-blocking asynchronous I/O and execute code quickly, making it a fundamental component. The event-driven design of the system enables efficient handling of concurrent processes, which is essential in a dynamic environment like a library management system. Complementing Node.js, Express.js, a basic but powerful online application framework, simplifies the building of RESTful APIs. The modular and extendable structure of the system integrates effortlessly with the changing demands of a project.

At the core of the data layer, MongoDB, a NoSQL database, presents a paradigm change with its schema-less architecture. This is very beneficial in the context of a library management system, especially when dealing with a wide range of different and constantly changing data. MongoDB's ability to manage many forms of data, ranging from literary works and writers to user profiles, establishes it as a flexible and scalable option.

The JSON format, a common ground for both Node.js and MongoDB, streamlines data transmission, boosting the interoperability between the server and client components. The real-time capabilities of Node.js enable the implementation of features like as live updates and alerts, which enhance the user experience.

❖ Folder structure for a Node JS project

For a Node JS project to be readable, scalable, and maintainable, a well-thought-out folder structure is essential. Effective management of code, configurations, modules, and other assets is aided by a clear framework. We will learn about the Node JS project's folder structure in this tutorial. We will get familiar with every directory seen in the Node JS project's folder structure and files.

- Required conditions

In order to comprehend a Node JS project's folder structure, make sure that:

+ A basic understanding of npm and Node JS
+ An IDE or code editor with Node JS installed on your computer

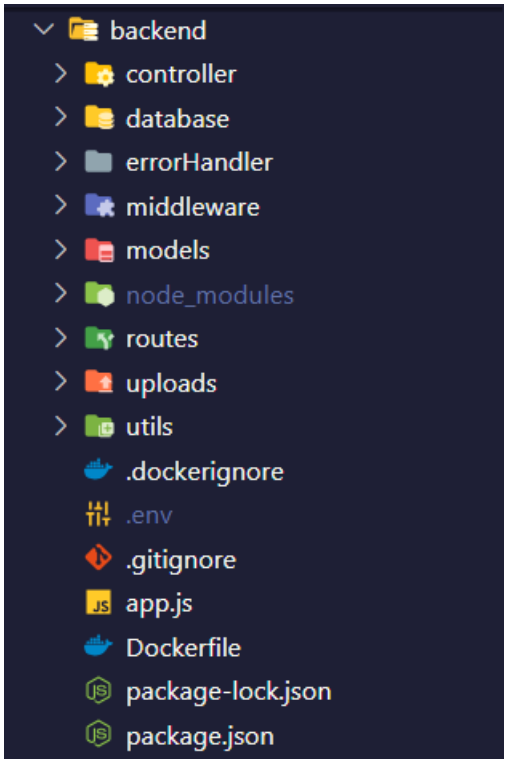


Figure 22: Node project folder structure

The files and folders that are present in the directory are contained in the folder structure. There are several files and directories, such as: app.js, server.js, package-lock.json, package.json, readme.md, controller, database, middleware, models, routes, utils , .env , node_modules.

- Guide for BACKEND Setup:

+ Open the terminal and go to the path where you want to create the project and create a folder with your project name.

Figure 23: create the project.

+ Run the command npm init to initialized the node project. This will create the package.json file

```
PS C:\Users\DELL\Desktop\library_mern> cd .\THESIS_ONLINE-LIBRARY-MANAGEMENT\
PS C:\Users\DELL\Desktop\library_mern\THESIS_ONLINE-LIBRARY-MANAGEMENT> npm init
```

Figure 24: create the package.json file.

+ Install dependencies like express, nodemon etc. This will create package-lock.json file and node_modules folder.

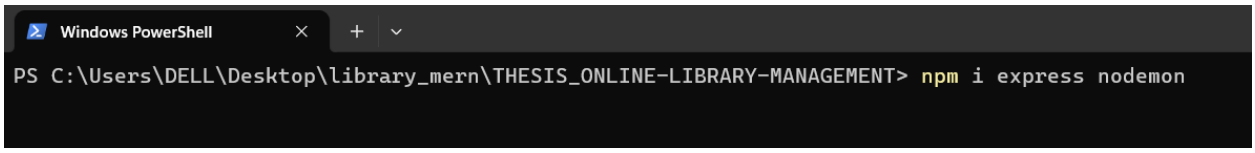


Figure 25: create package-lock.json file and node_modules.

+ Run the command git-init to initialize the git in the project. This will add .gitignore file.

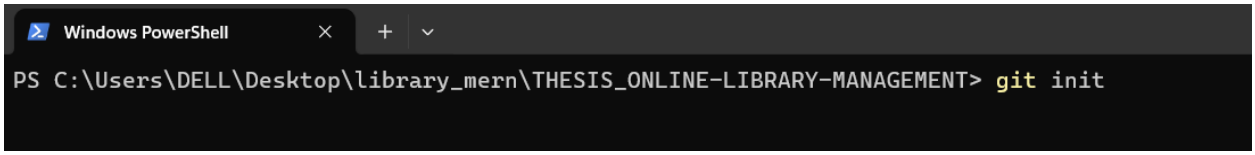


Figure 26: add .gitignore file.

+ Make a file with the .env extension that will hold the project's credentials and sensitive data.

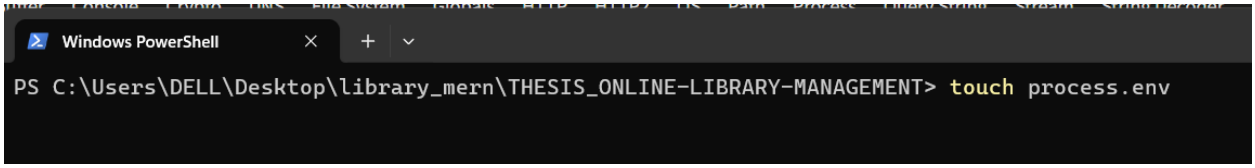


Figure 27: create .env file

+ Setup .env | Database CONNECTION_URL, Port Settings, JWT INFORMATION, EMAIL VERIFICATION.

The CONNECTION_URL for the database is defined in this phase, enabling simple customization and flexibility. Setting the port in the .env file also guarantees consistency across environments and improves portability.

To enhance the security of your application and integrate crucial features like authentication and email verification, we'll include extra environment variables in the .env file.

JWT Configuration: JWT_SECRET:
This variable represents the secret key used to sign JWTs, assuring their integrity and legitimacy.

JWT_REFRESH_SECRET: Like the JWT_SECRET, this variable is devoted to the secret key used for signing refresh tokens.

JWT_LIFE: Defines the expiry period for the access tokens. After this time, users will need to re-authenticate.

JWT_REFRESH_LIFE: Specifies the expiry period for refresh tokens, enabling a regulated and secure token renewal process.

Email Verification Configuration:
EMAIL_USERNAME: Represents the username or email address used for sending verification emails.

EMAIL_PASSWORD: Enter your Gmail Account 'APP Password' not normal login password .App Password can be generated in google account after unlocking 2factor authentication

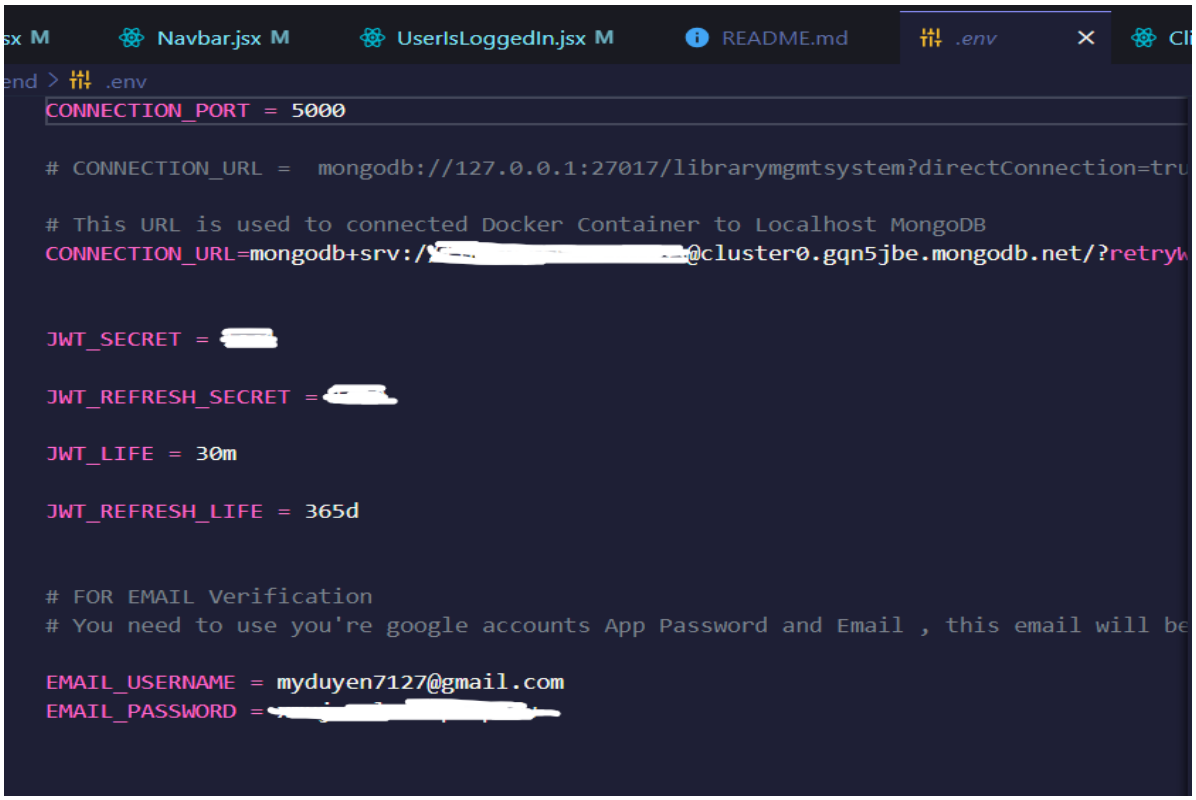


Figure 28: set up .env file.

+ Create server: The basis of the backend begins with the development of a server. Within this server, we establish the fundamental components responsible for processing API requests and answers.

- Models: The model captures the structure of the database collections. It comprises schema definition and data validation rules. This protects the integrity and consistency of the data contained in the database.

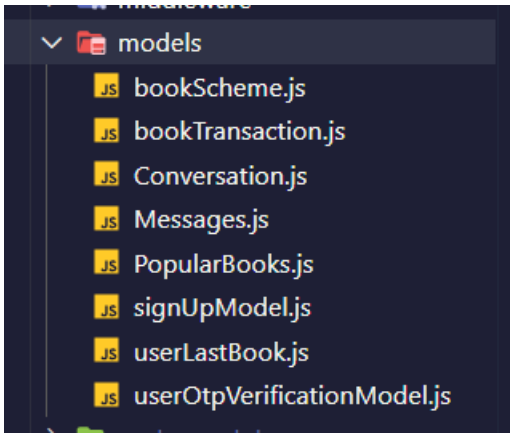
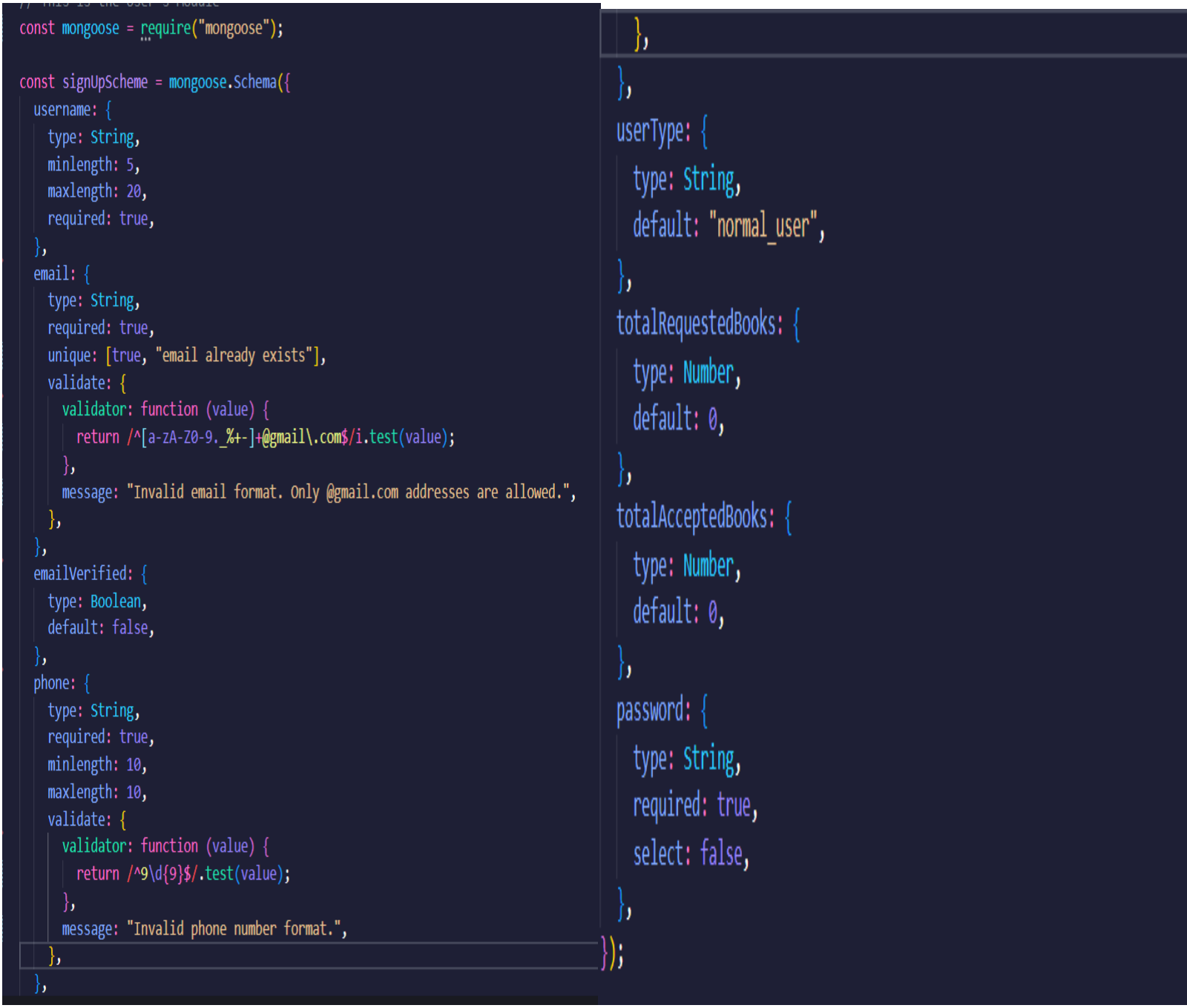


Figure 29: Create Models server.

Example of user details model , have some model attributes:

- Username: The username, a fundamental attribute, serves as a unique identifier for each user.
- Email: The email attribute is a cornerstone for user communication and authentication.
- Email Verification Status (emailVerified): A binary indication showing if the user has confirmed their email. This property adds to the application's security and guarantees that only authorized users may access particular functions.
- Phone: One further layer of contact information is added by the phone property. Even though it's optional, it may be helpful for alerts or two-factor authentication and creates channels of communication outside of standard email.
- User Type (userType): This feature groups individuals according to their rights or responsibilities in the system. It establishes the framework for putting role-based access control into practice, making sure that various user categories have customized rights and access levels.
- Total Requested Books (totalRequestedBooks): The number of books a user has requested via the system is reflected in this numerical characteristic. It helps with resource allocation and book recommendation algorithm improvement by offering insights into user participation and interests.
- Total Accepted Books (totalAcceptedBooks): Similar to total requested books, this attribute signifies the count of books a user has successfully obtained or had their requests accepted. It contributes to a comprehensive understanding of user interactions and satisfaction.
- Password: The password attribute encapsulates a secure hashed representation of the user's password. Implementing secure password storage practices is paramount for user account protection and privacy.



```
const mongoose = require("mongoose");

const signUpScheme = mongoose.Schema({
  username: {
    type: String,
    minlength: 5,
    maxlength: 20,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: [true, "email already exists"],
    validate: {
      validator: function (value) {
        return /^[a-zA-Z0-9._%+-]+@gmail\.com$/i.test(value);
      },
      message: "Invalid email format. Only @gmail.com addresses are allowed.",
    },
  },
  emailVerified: {
    type: Boolean,
    default: false,
  },
  phone: {
    type: String,
    required: true,
    minlength: 10,
    maxlength: 10,
    validate: {
      validator: function (value) {
        return /^9\d{9}$/.test(value);
      },
      message: "Invalid phone number format.",
    },
  },
  },
  userType: {
    type: String,
    default: "normal_user",
  },
  totalRequestedBooks: {
    type: Number,
    default: 0,
  },
  totalAcceptedBooks: {
    type: Number,
    default: 0,
  },
  password: {
    type: String,
    required: true,
    select: false,
  },
});
```

Figure 30: User Model

- Controller: Controllers are responsible for implementing the functionality associated with each route. They operate as the brains behind the operations, executing the required logic depending on the incoming requests. Handles the Functionality of all the Routes

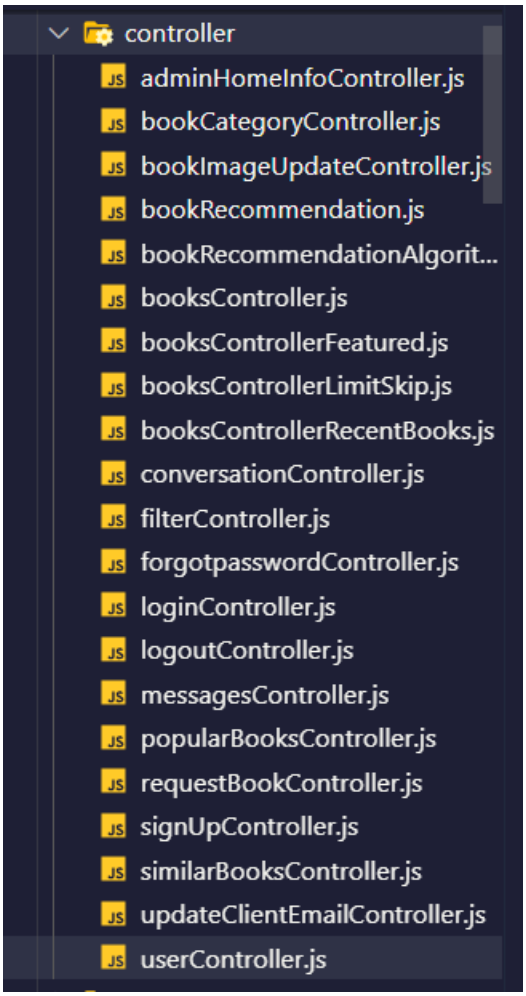


Figure 31: Create controller

- The establishment of a user controller is a crucial element in constructing a resilient and efficient online application. The getAllUsers, getSingleUser, postSingleUser, and patchUserDetail methods jointly provide the retrieval, creation, and updating of user information, ensuring a smooth interaction between the frontend and backend components of the application. By proficiently using these techniques, developers may guarantee a seamless and successful user management system inside their online applications.

```
// Fetch all USERS data + check book return status and FINE Charge
const getAllUsers = async (req, res) => {
  const result = await UserModel.find({ userType: 'normal_user' })

  res
    .status(200)
    .json({ success: true, totalHits: result.length, data: result })
}

// Fetch single user data + book transactions using /Params
const getSingleUser = async (req, res) => {
  const { userId } = req.params

  const getUserData = await UserModel.findById(userId)

  const getUserBookTransaction = await BookTransactionSchema.find({
    userId,
    issueStatus: 'ACCEPTED',
    // PENDING & ACCEPTED only issueStatus fetch
    // issueStatus: { $in: ['PENDING', 'ACCEPTED'] },
  })

  // Fetch all 3 Status to show users - ACCEPTED / PENDING / CANCELLED
  const getAllUserBookTransaction = await BookTransactionSchema.find({
    userId,
  })

  res.status(200).json({
```

Figure 32: User Controller

- Routes: Routes play a critical role in determining the API endpoints (Handles all the API routing). They operate as the gateway, forwarding incoming requests to the relevant controllers.

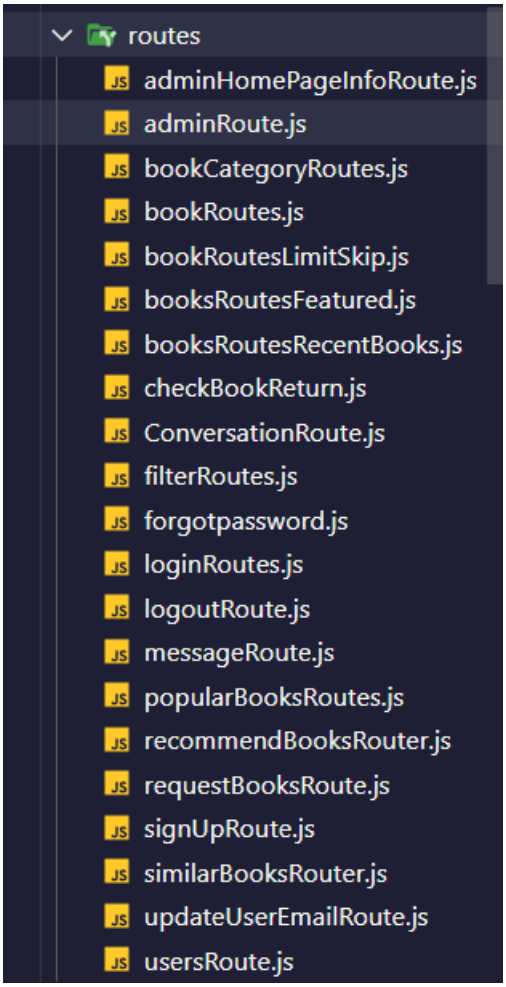


Figure 33: create Routes

- The individuals using the systemThe route is essential for overseeing user-related processes in the backend of a web application. The getAllUsers, getSingleUser, postSingleUser, and patchUserDetail APIs jointly provide an extensive range of features for getting, generating, and updating user information. Developers may construct a strong and user-friendly backend system by efficiently implementing these routes and their corresponding controller methods. This will allow for smooth integration with the frontend components of the application.

usersRoute.js

backend > routes > usersRoute.js > ...

```
1   const express = require('express')
2   const userRouter = express.Router()
3
4   // Middleware
5   const adminAuthorization = require('../middleware/adminAuth')
6
7   const {
8     getAllUsers,
9     getSingleUser,
10    postSingleUser,
11    patchUserDetail,
12  } = require('../controller/userController')
13
14  userRouter
15    .route('/')
16    .get(adminAuthorization, getAllUsers)
17    .post(postSingleUser)
18    .patch(patchUserDetail)
19
20  userRouter.route('/:userId').get(getSingleUser)
21
22  module.exports = userRouter
```

Figure 34: users Route

- Connect Mongo Database
- Log in or register an account on the mongoDB homepage:

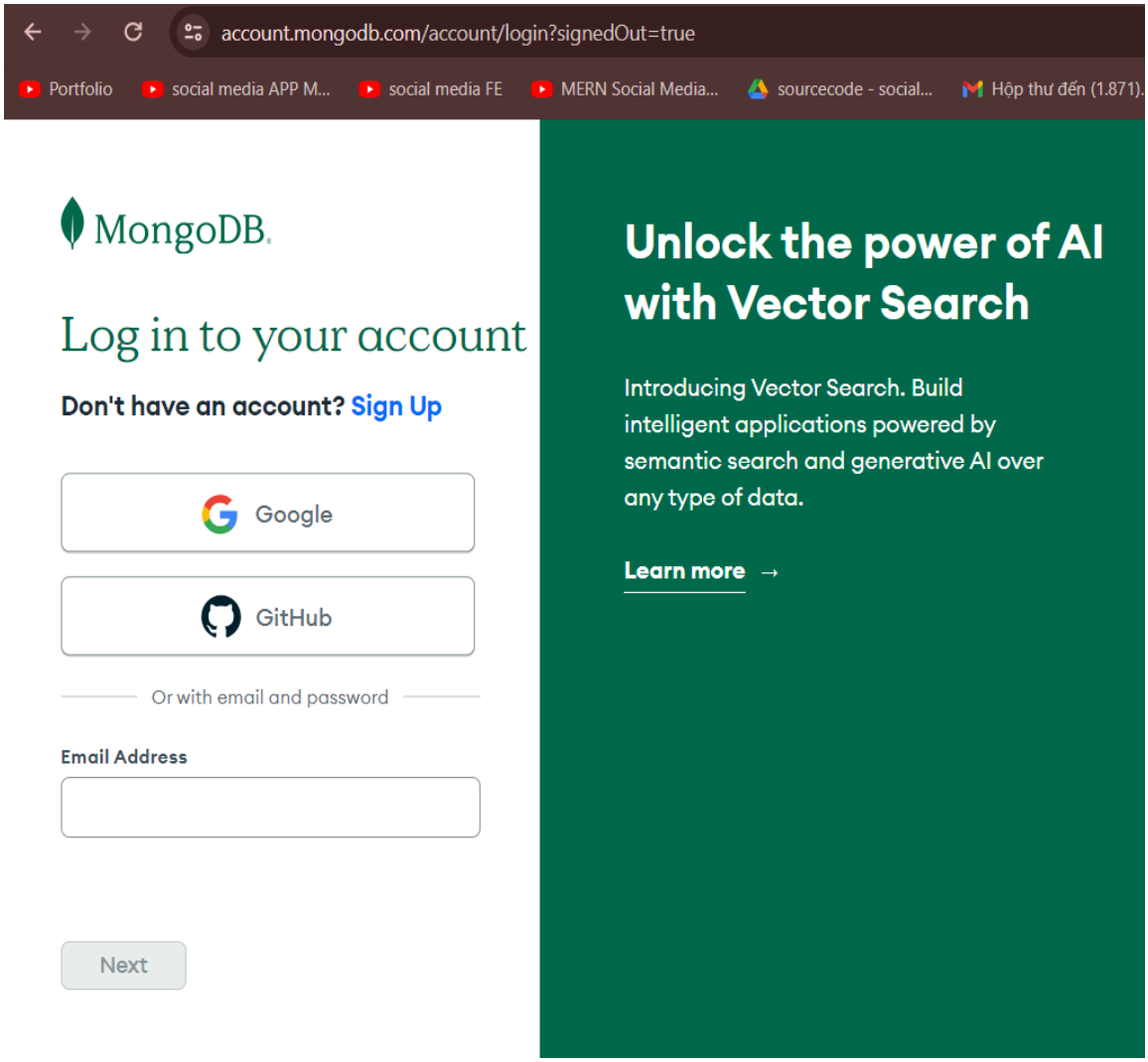


Figure 35: login MongoDB

- Create a database to connect to mongoDB drive:

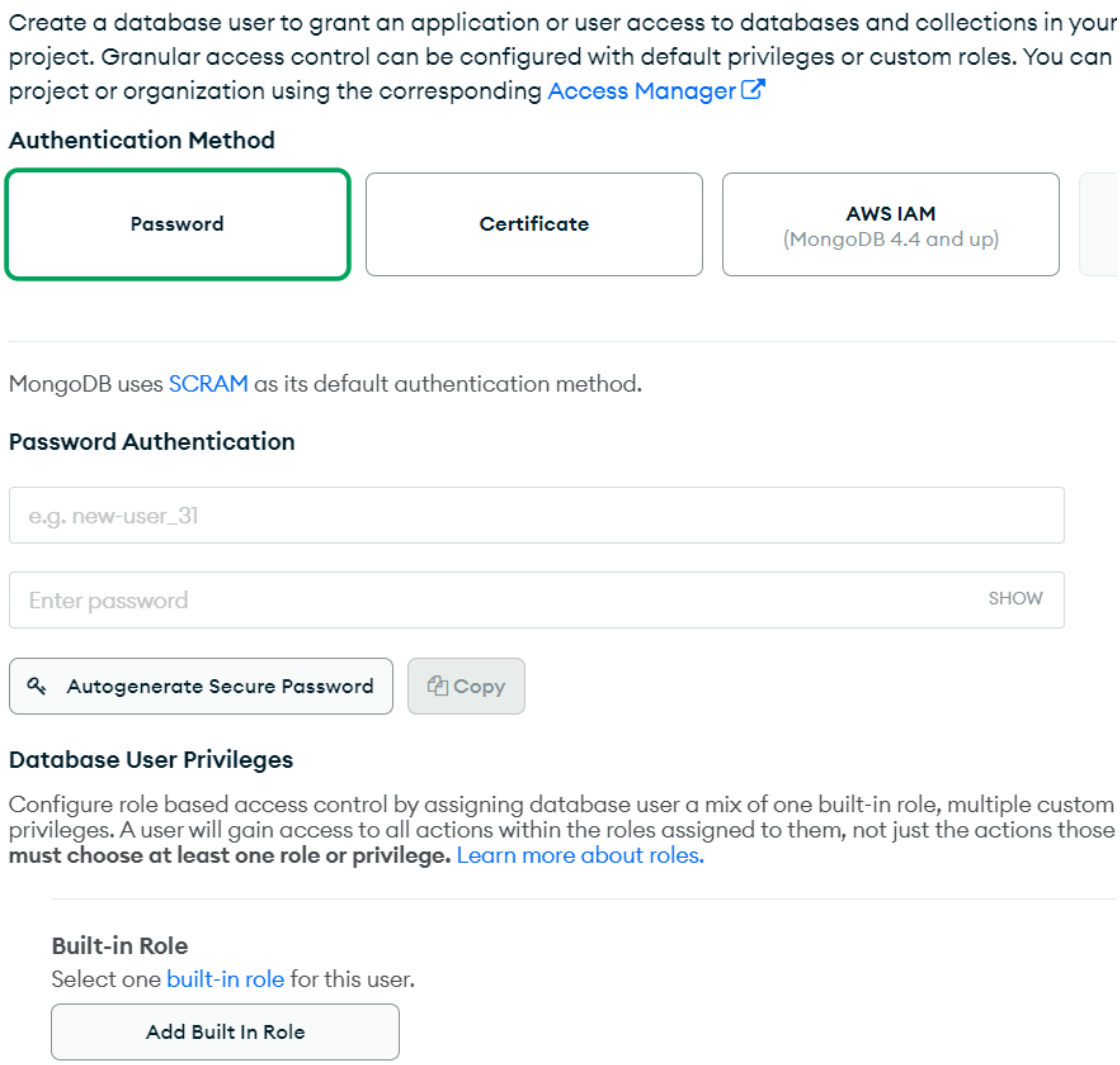


Figure 36: create a database Mongo

- Add connection string into the application code and saved in **.env** file. **<username>** and **<password>** are information when creating a database

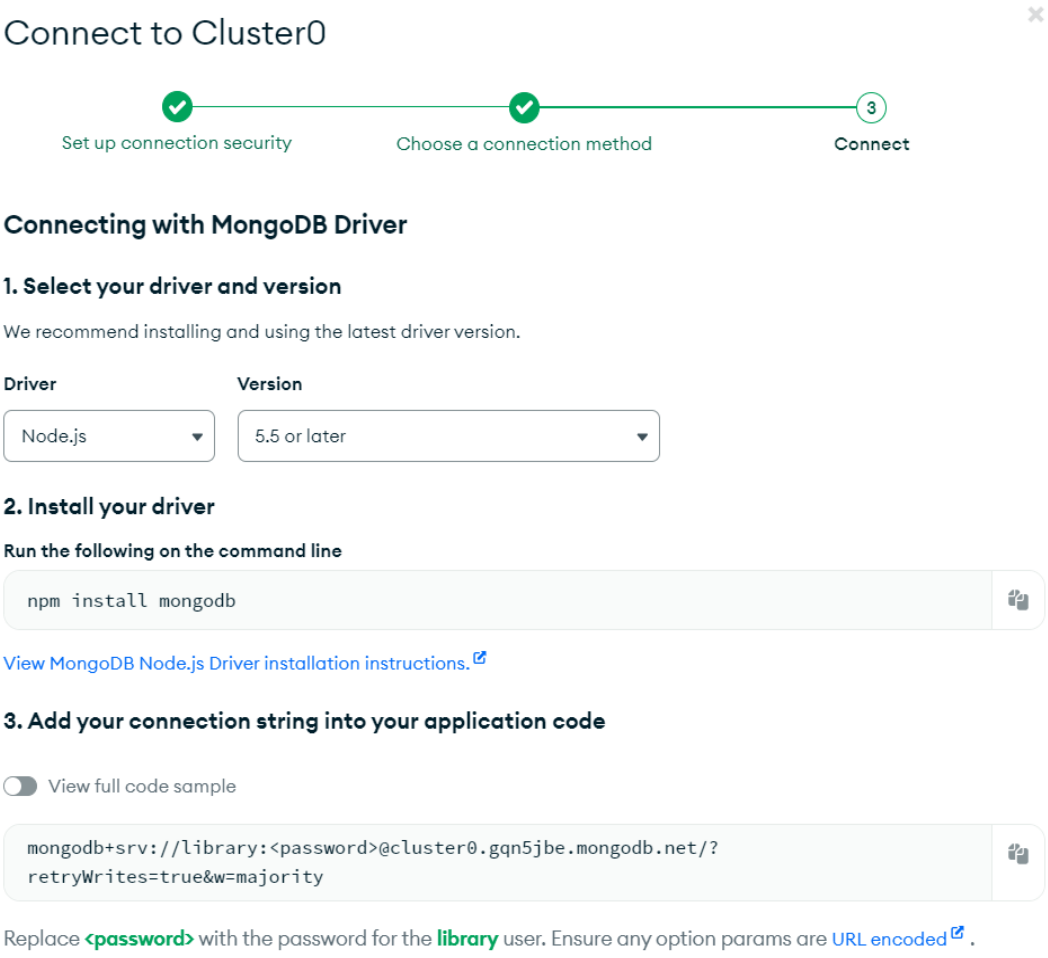


Figure 37: Add connection string

- Setup to connect the backend to MongoDB:

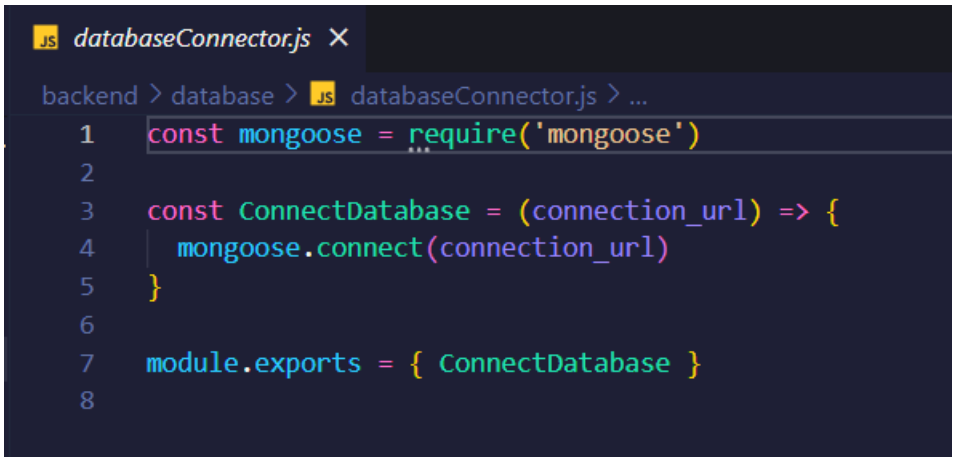


Figure 38: connect BE to MongoDB

- Setup **POSTMAN** | Collections && Global Variables

- The Postman application is an essential tool for both testing and documenting APIs. Through the creation of collections and the definition of global variables, the testing process is made more efficient. Collections serve to organize API requests that are connected, while global variables facilitate the administration of dynamic data throughout the requests.

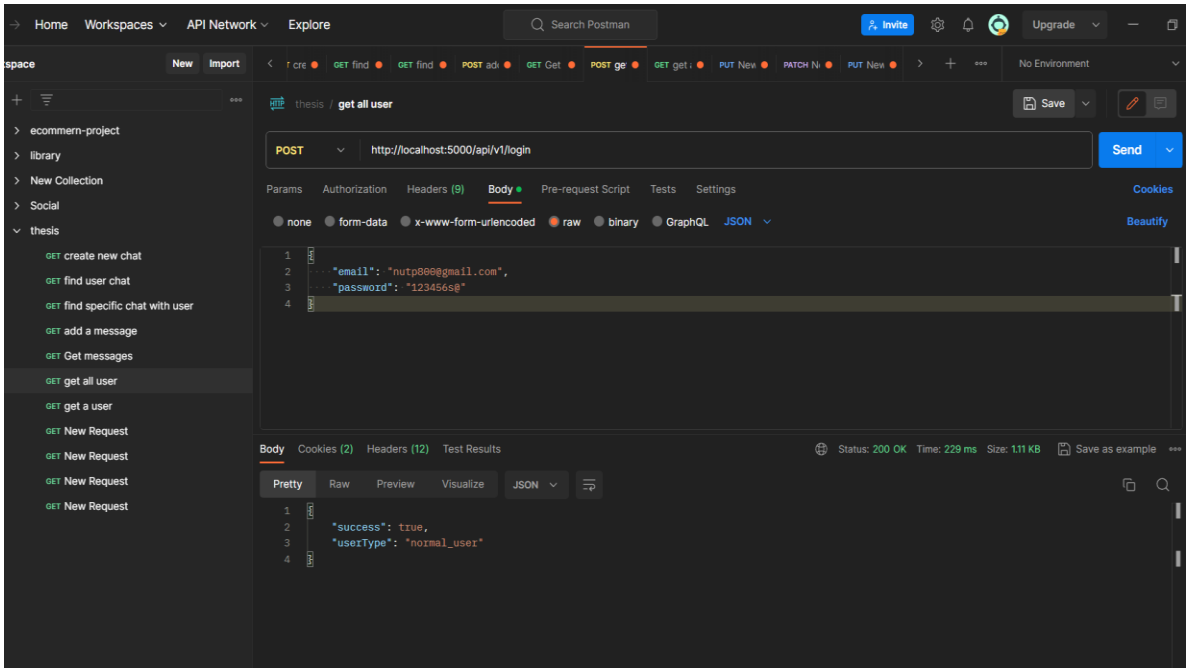


Figure 39: Setup Postman

- API Reference (BOOKS):

  + Get all books: GET /api/v1/books

  + Post Book: POST /api/v1/books

  + Get Single book: GET /api/v1/books/:id

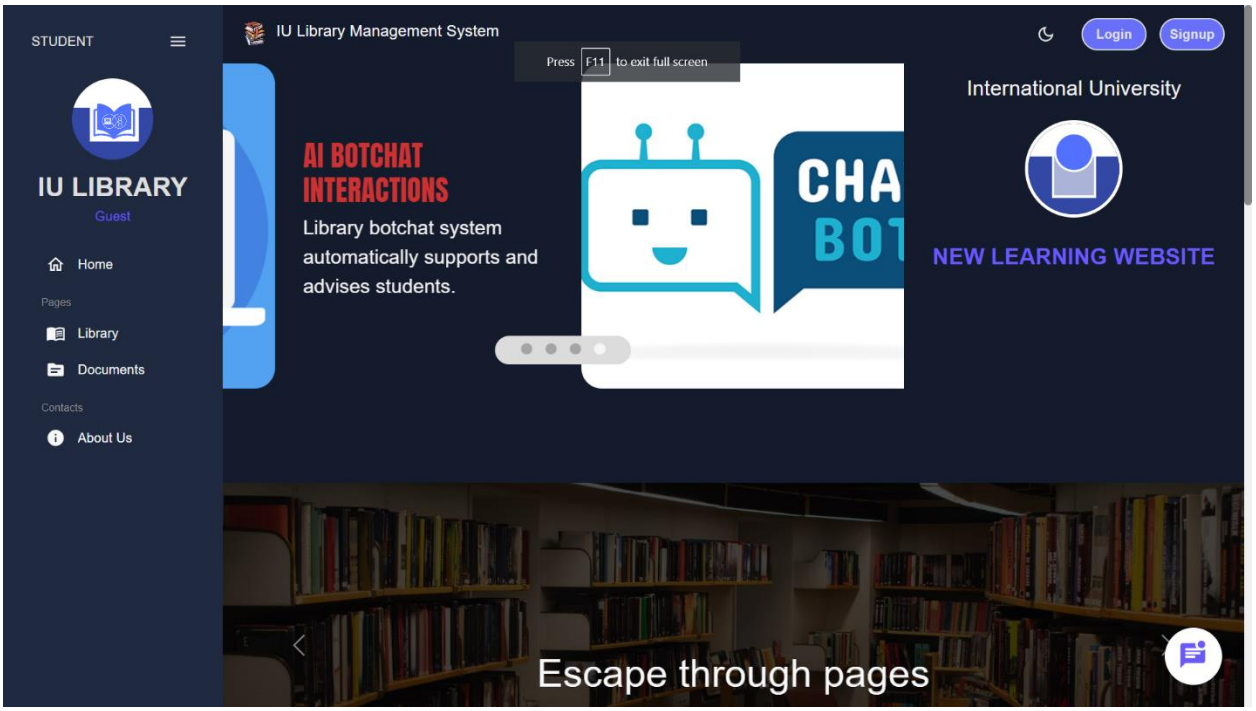
  + Update Single book: PATCH /api/v1/books/:id

  + Delete Single book: DELETE /api/v1/books/:id

4.2 Result

4.2.1 Guest's home page

- This is the guest home page interface, on the nav bar there are 2 buttons to register and log in. On the left sidebar there is guest information



Fgiure 40: Guest home page

- The guest's home page displays latest and featured books:

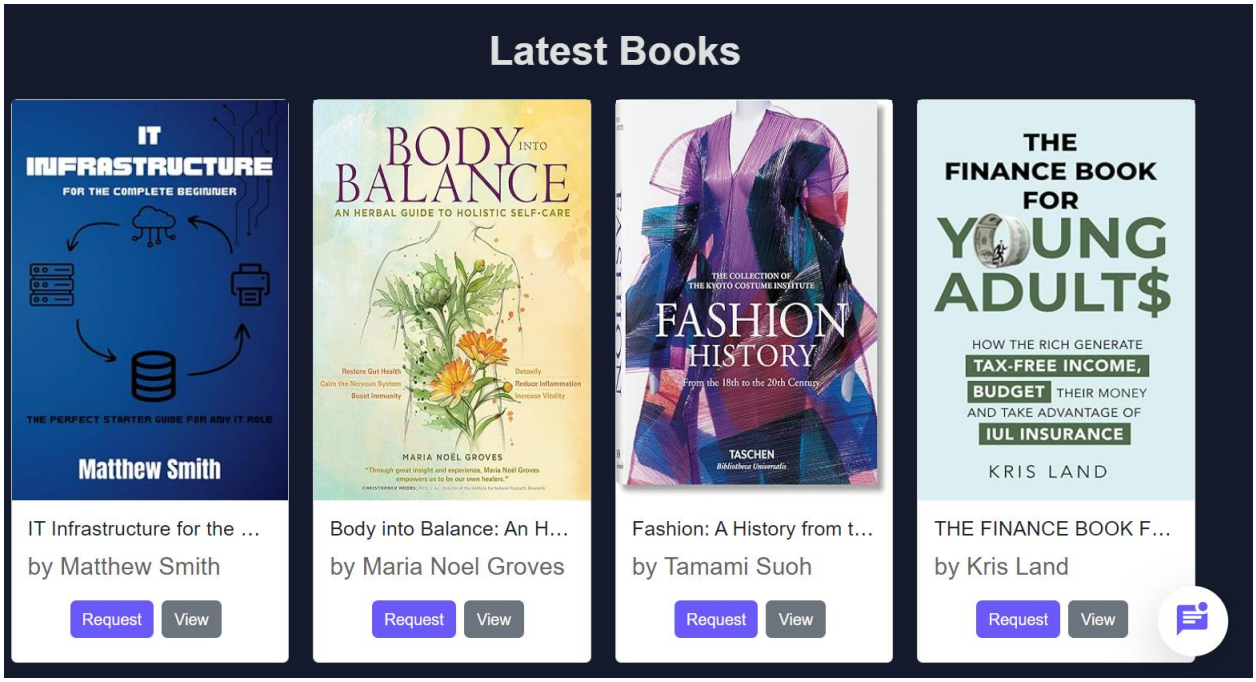


Figure 41: Latest books

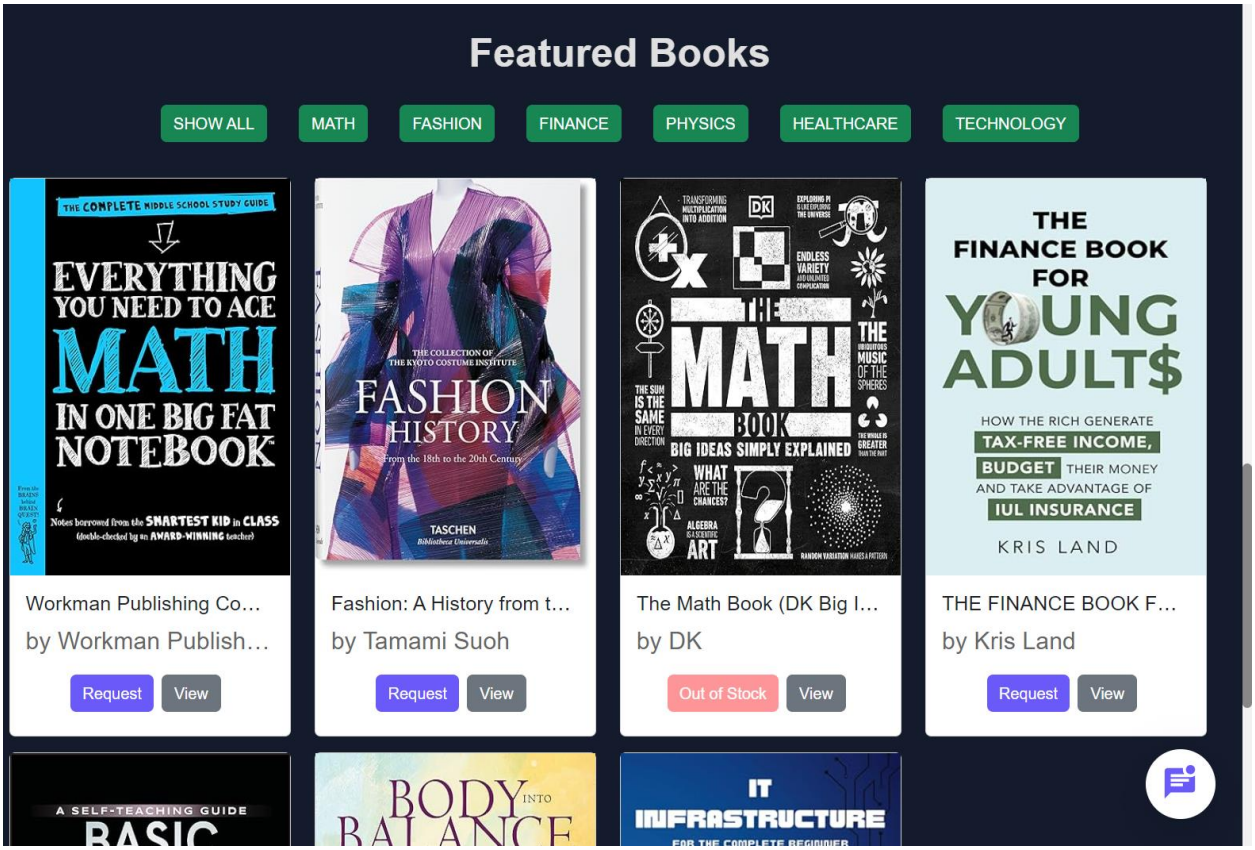


Figure 42: Featured Books

4.2.2 Student's home page

- On the **student's home page**, the nav bar has been replaced with notification icons and settings icons to navigate to the profile page or log out from the website. In addition, on the home page, students can see **recommended books** that guests cannot have and learn through displaying the latest books and featured books. The left sidebar displays student email information when logged in

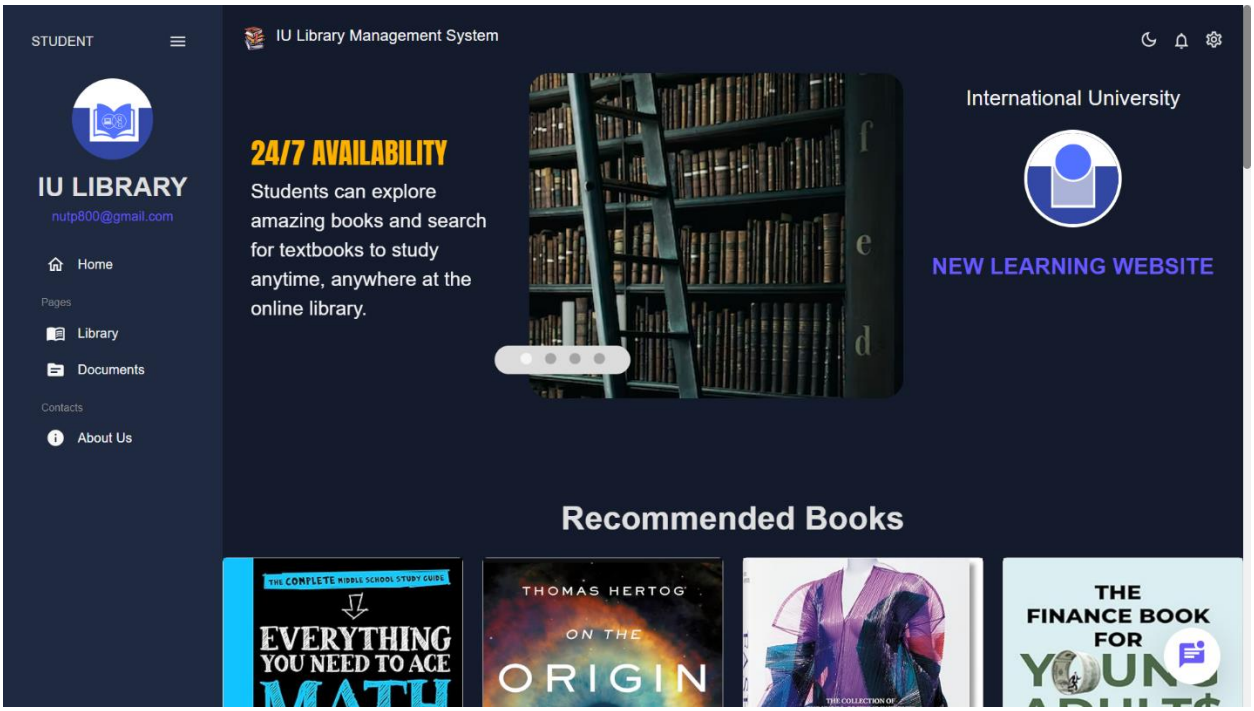


Figure 43: Student home page

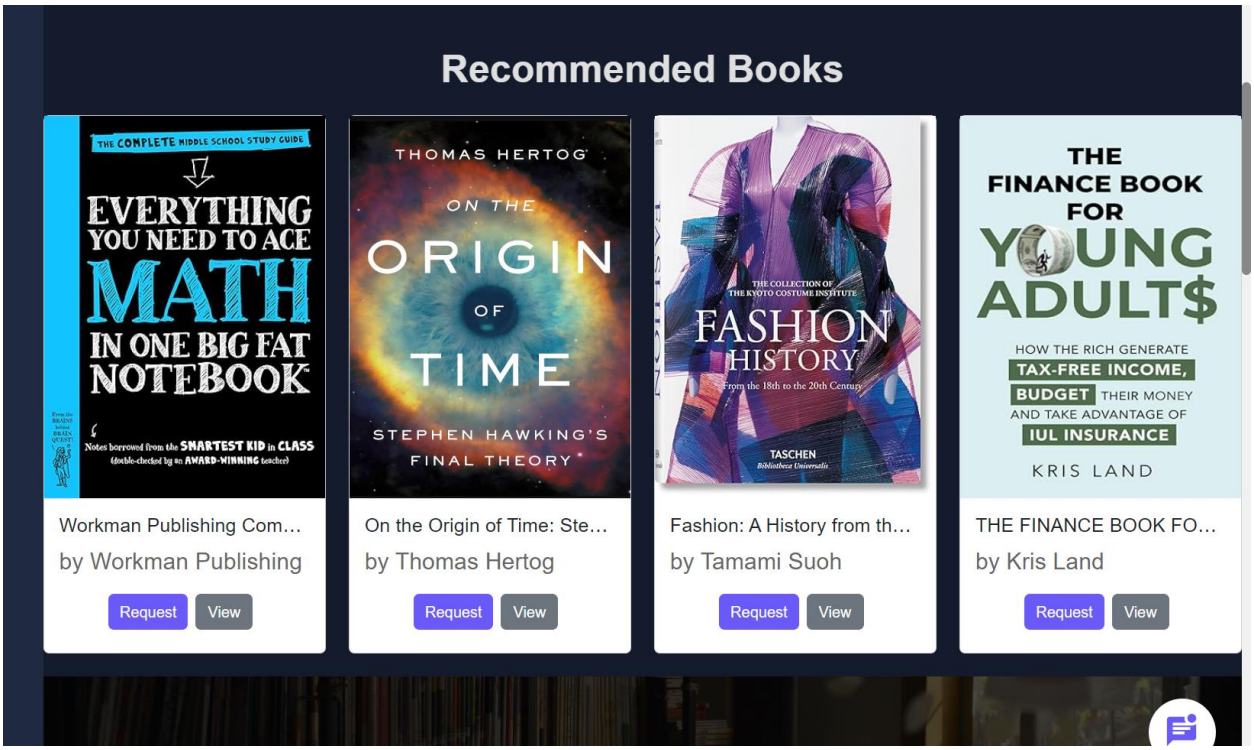


Figure 44: Recommended Books

4.2.3 Login Page:

- This is the **login page** interface. On this page, if the guest has a previous account, they can fill in their email and password to log in and be navigated to the home page for students. The login page has a nav bar containing website services and contact information and a back to guest home page button. Below the login form, there are two forms: to retrieve a **forgotten password** and to **register an account.**

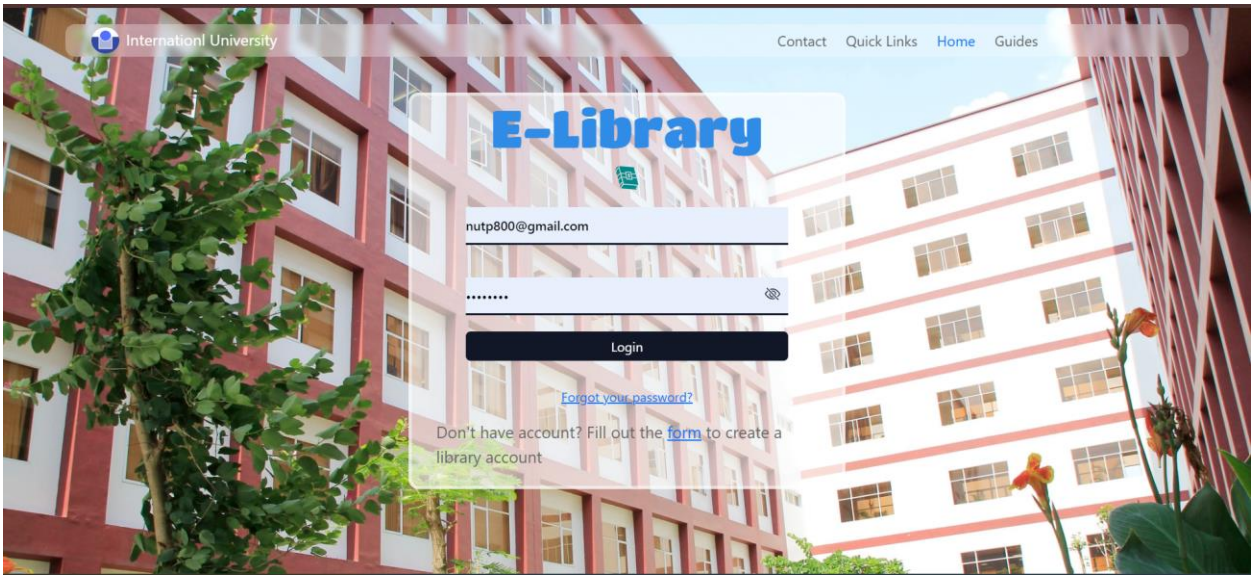


Figure 45: Login page.

4.2.4 Signup Page:

- This is the registration page interface, guests must fill in Student name, email (Gmail), phone, password and confirm password to register an account.

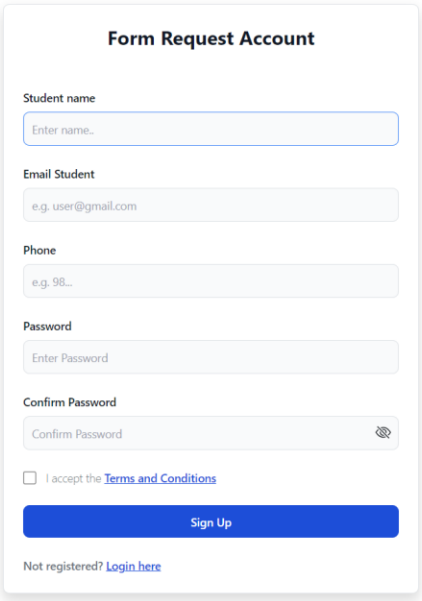


Figure 46: Signup page

4.2.5 Email Verification Page:

- This is the email verification page. When the guest has successfully registered with Gmail, the system's OTP code will be sent to the Gmail used to create it. Students enter that OTP code and successfully go to the login page to log in to the website.

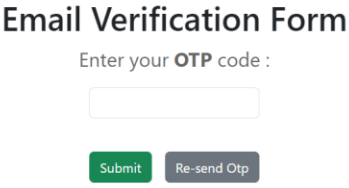


Figure 47: Email Verification page

4.2.6 Library Page:

- This is the library page interface, starting with **popular books** - these are the books that have been borrowed by the most students. Next is the library's **banner section** advertising outstanding book areas. Finally, there is a section on all book libraries, with a filter tool bar to help students quickly find the books they want. Each book has 2 **Request buttons** (can only be pressed if you are logged in) and a **View button** to view detailed information about that book.

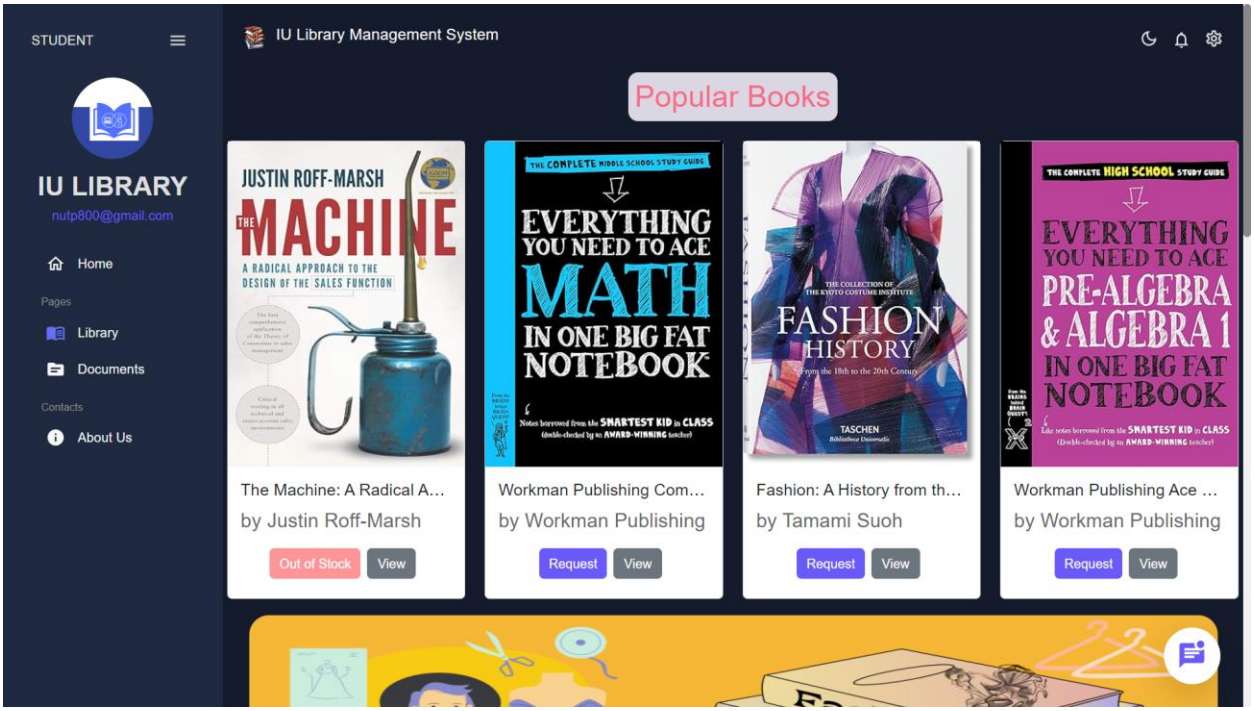


Figure 48: Library page

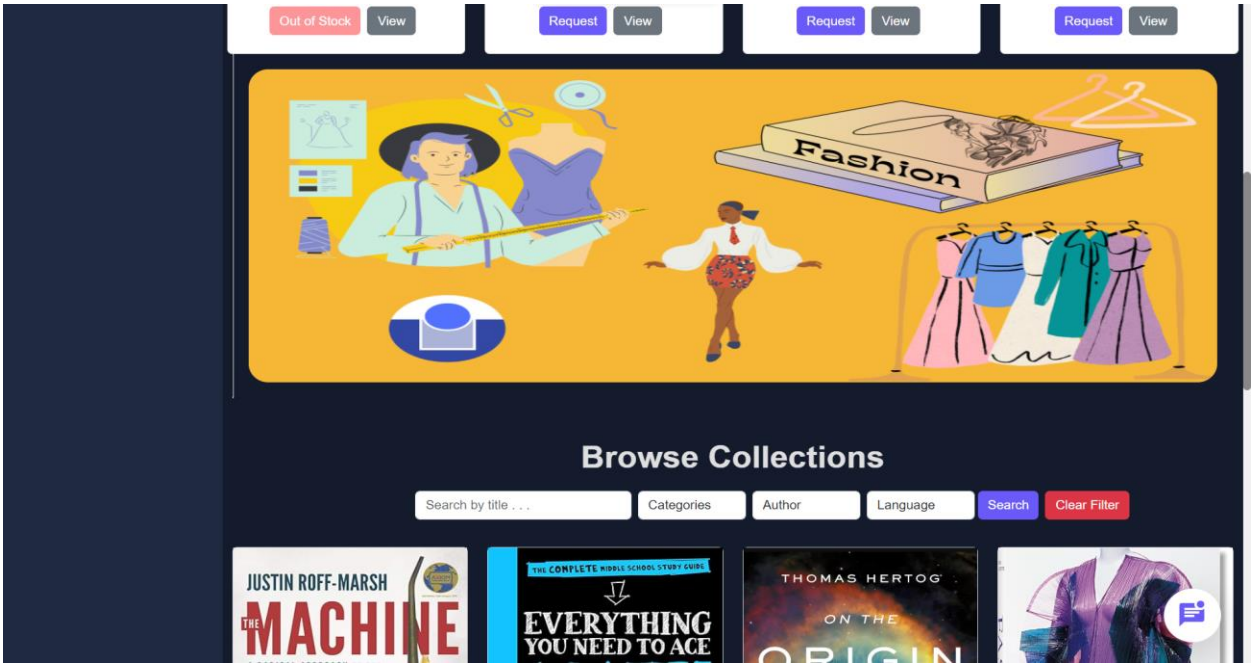


Figure 49: Banner in library page

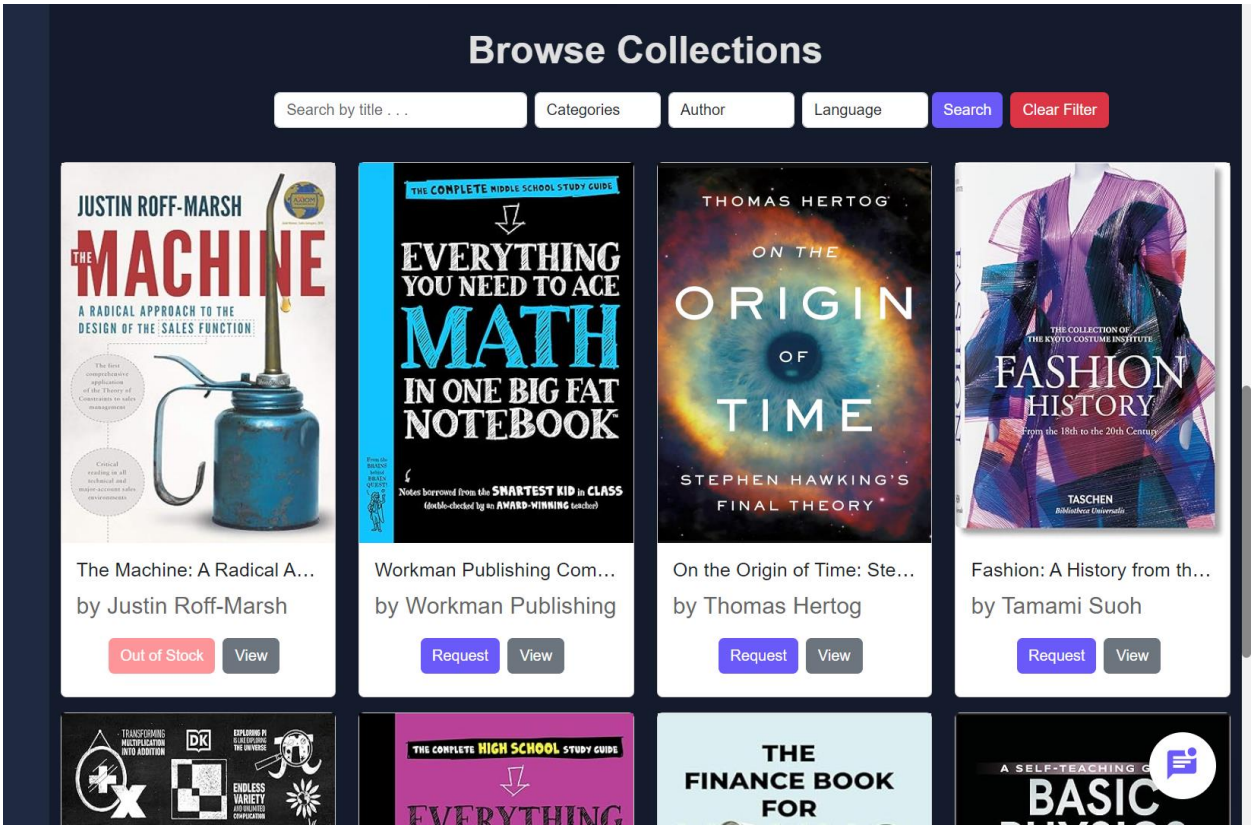


Figure 50: Browse Collections in library page

4.2.7 Book Details Page:

- This is the interface of the book detail information page. This page displays information about a book such as title, author, category, Language, Description. In the available status, if the book is in stock, students can request to borrow it, but if it is out of stock, they cannot borrow it. The section at the bottom of the page is books similar to the category of this book information.

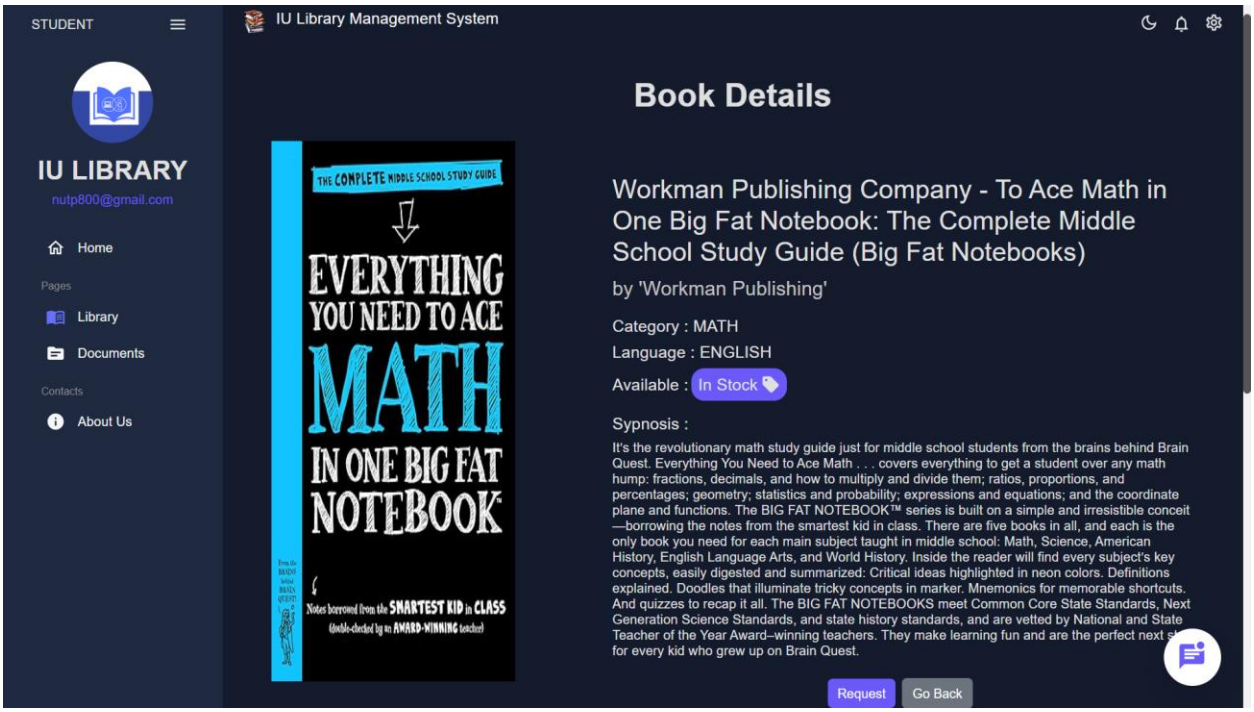


Figure 51: Book Details page (In Stock Status)

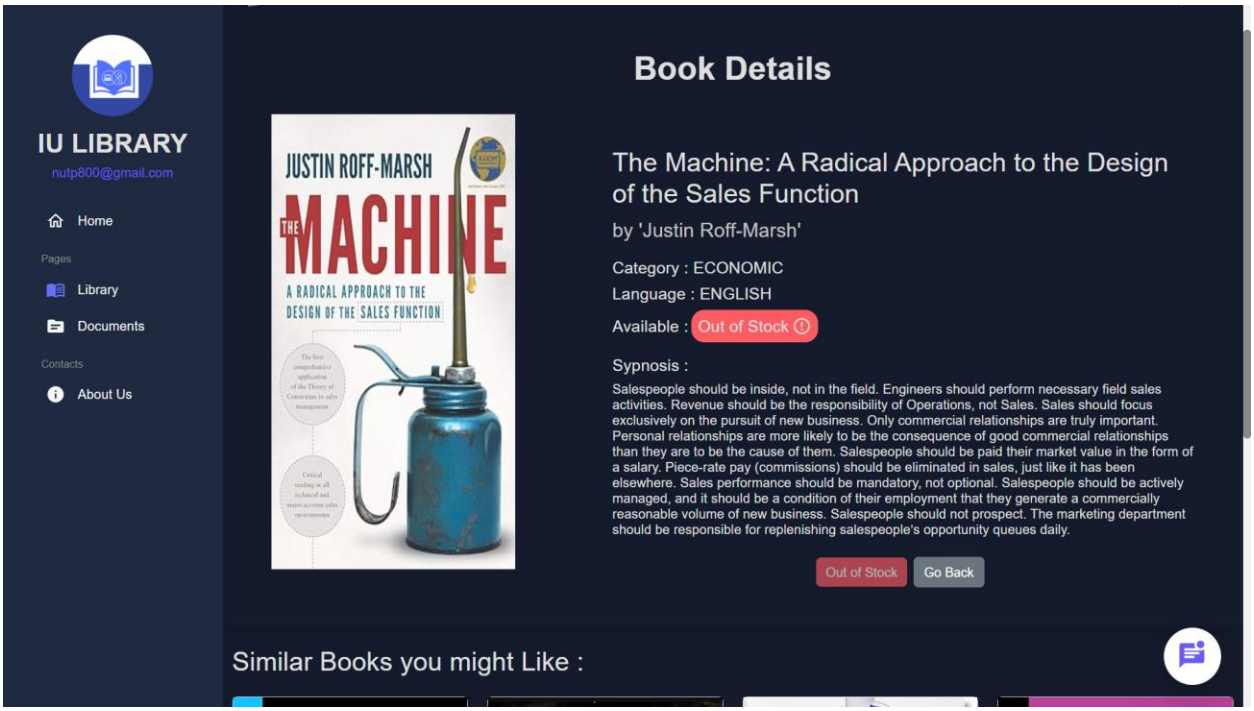


Figure 52: Book Details page (Out of Stock Status)

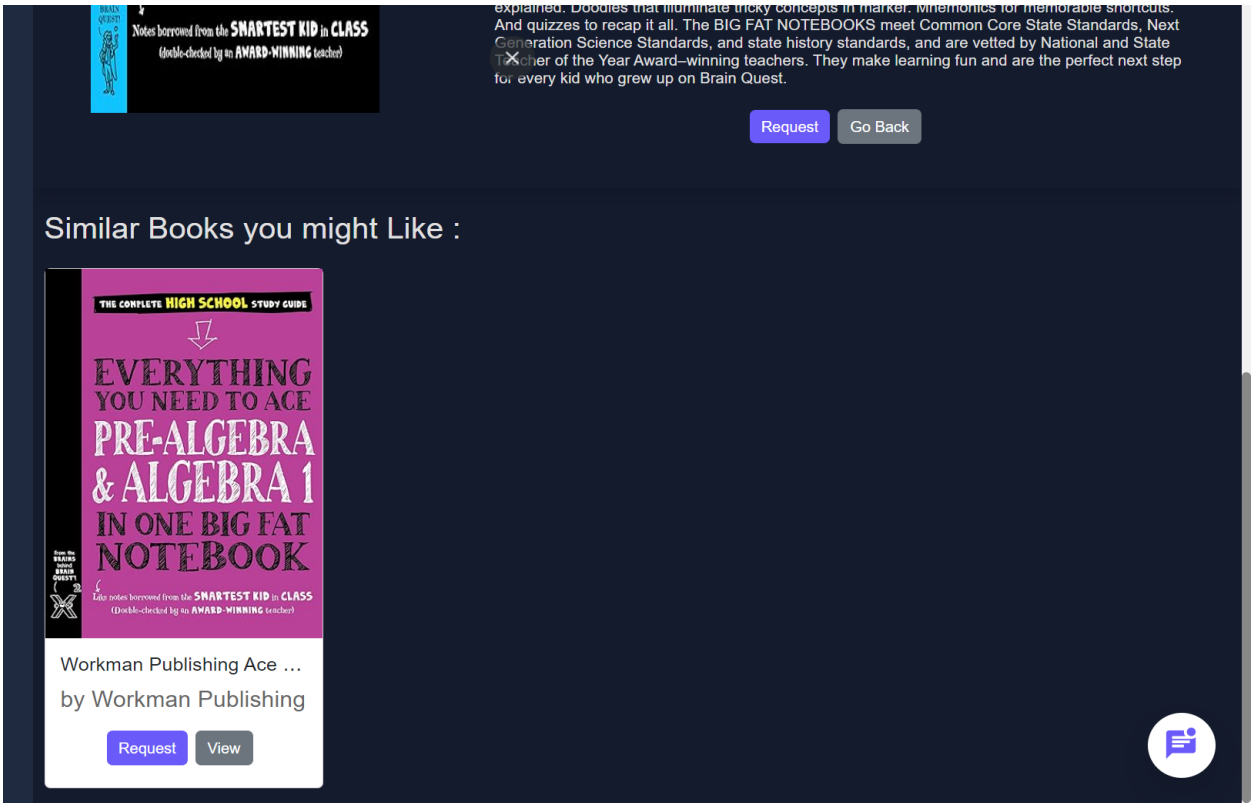


Figure 53: Similar Books in Book Details page

4.2.8 About us Page:

- This is the about us page interface, this page displays information and the process of building this online library website.

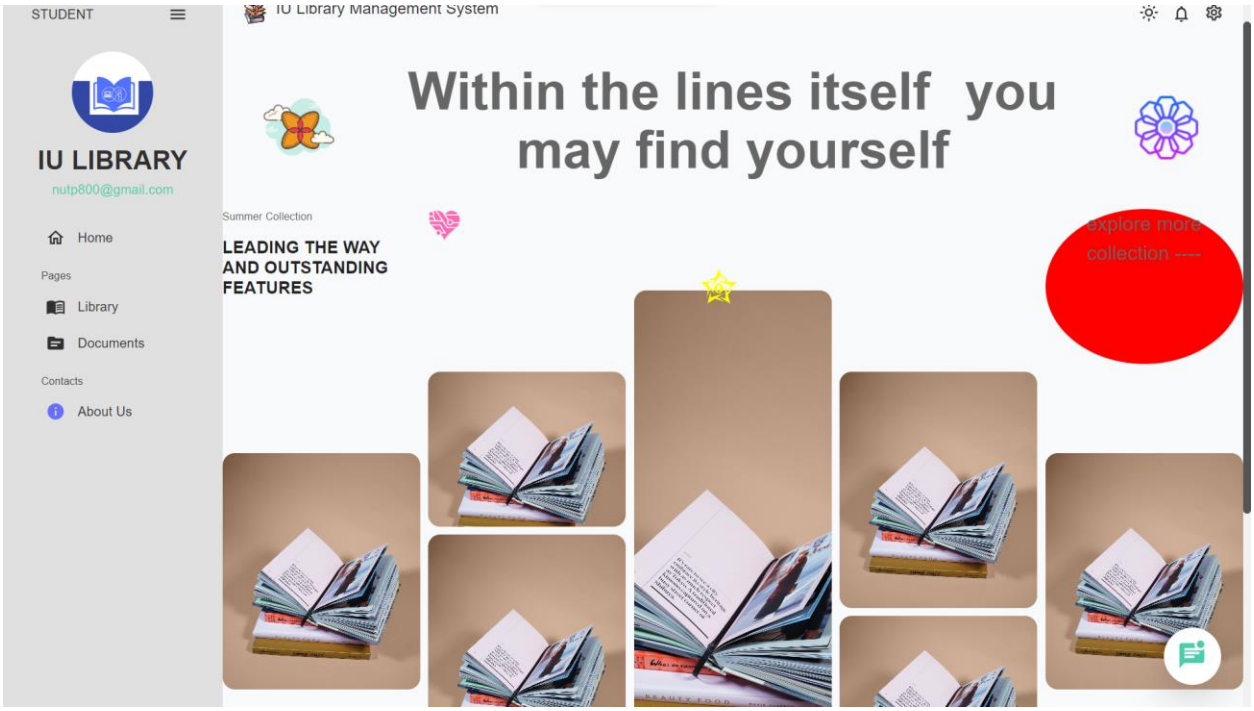


Figure 54: About us page

4.2.9 Profile Page:

- This is the profile page interface, this page has 2 buttons: history (showing the entire book borrowing history, system processing status, time to borrow and return books). In the My Details button is all information about this account including username, email, phone, total Book and password. Users on this page can update information.

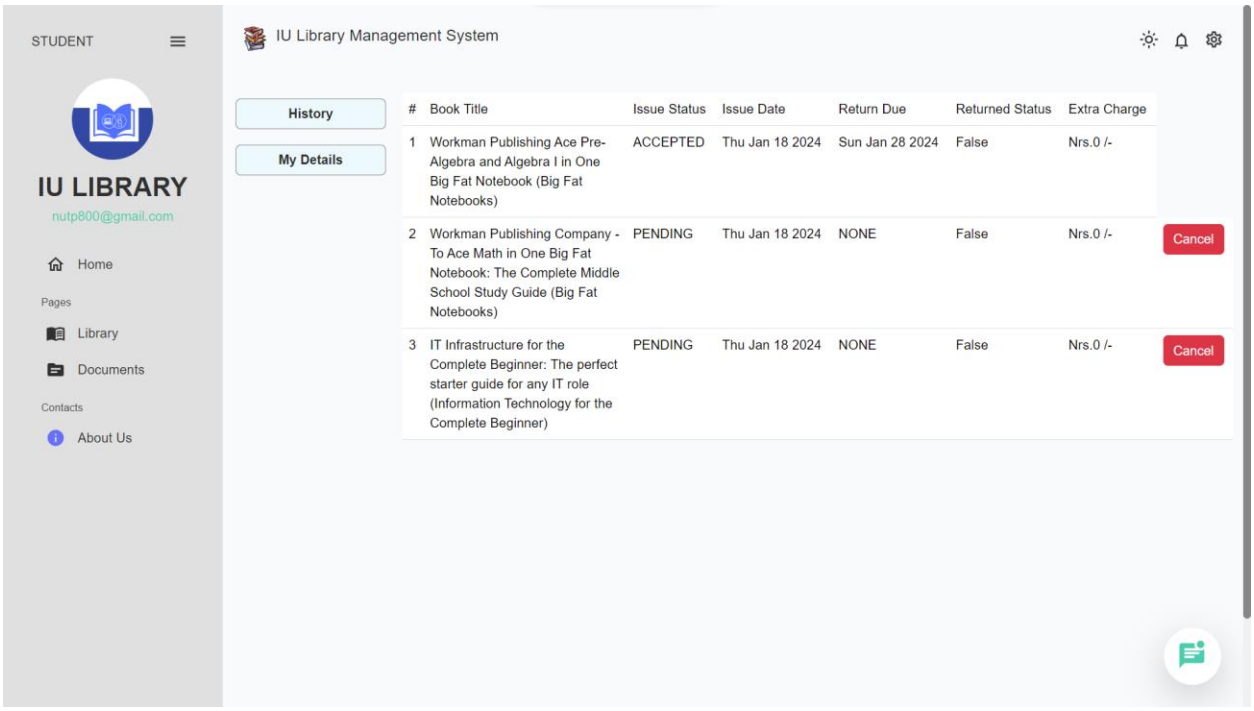


Figure 55: Profile page (History).

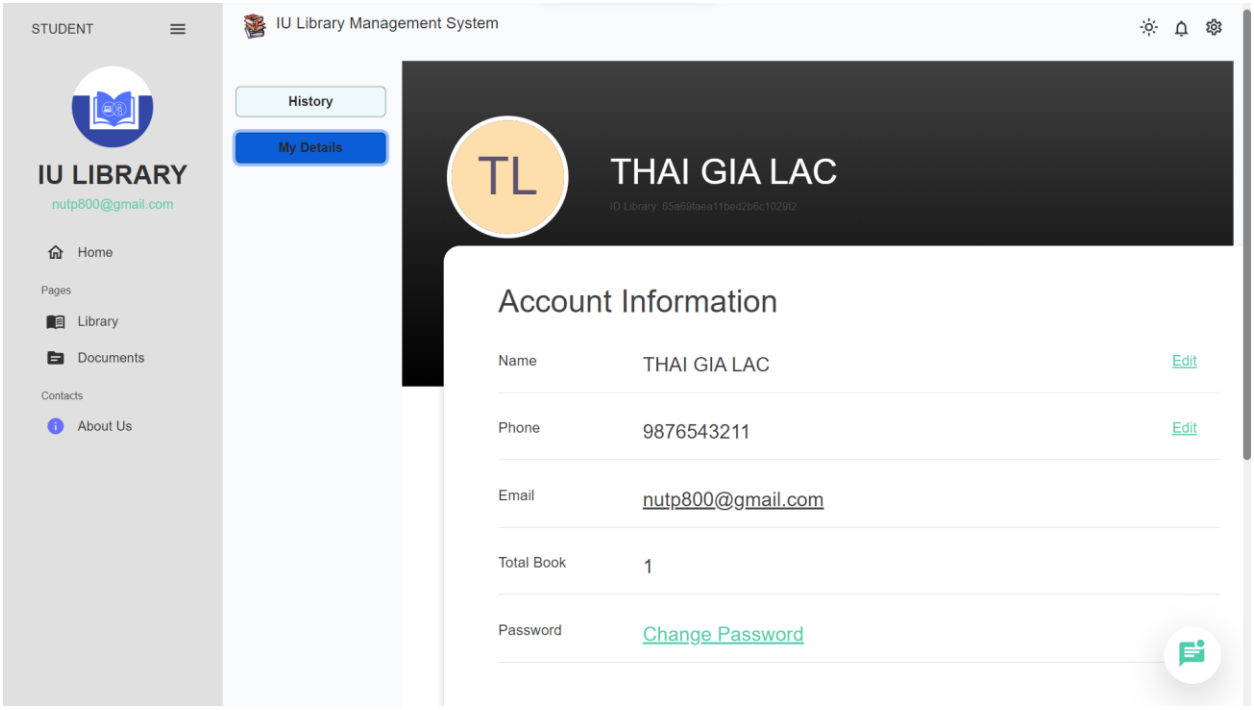


Figure 56: Profile page (My Details).

- When the user presses the edit button, a modal will appear to change the information, then the system will automatically update immediately.

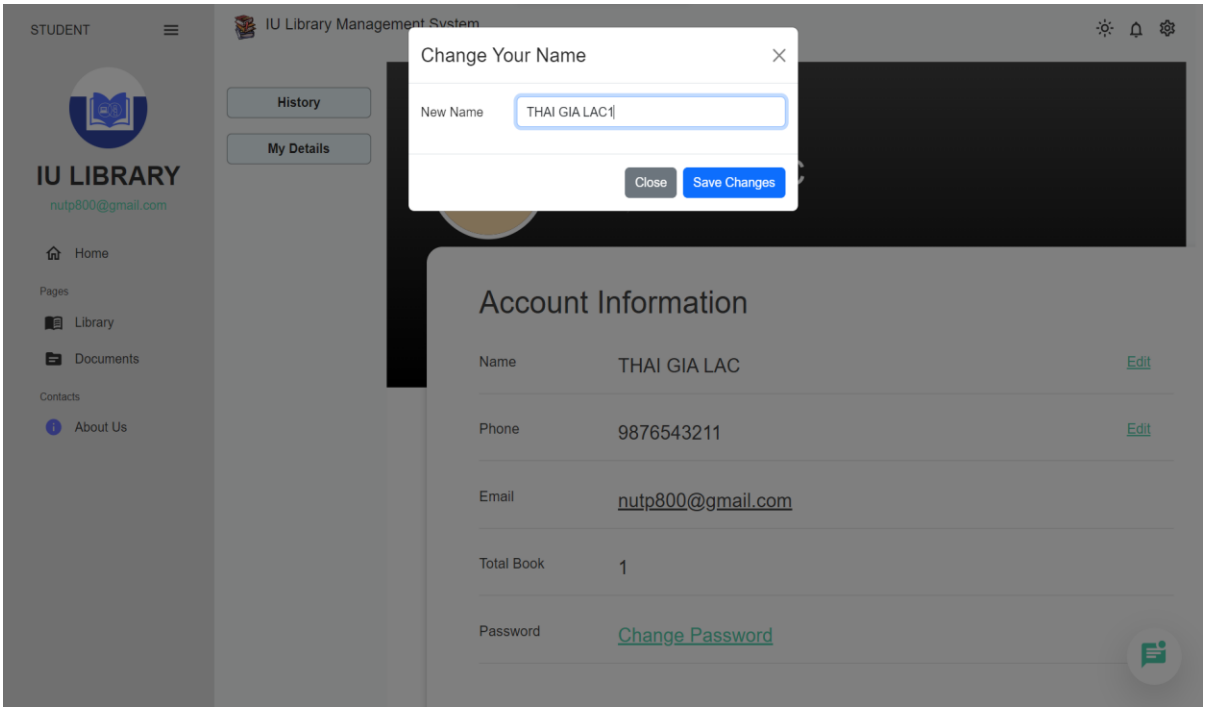


Figure 57: Fill Model update name

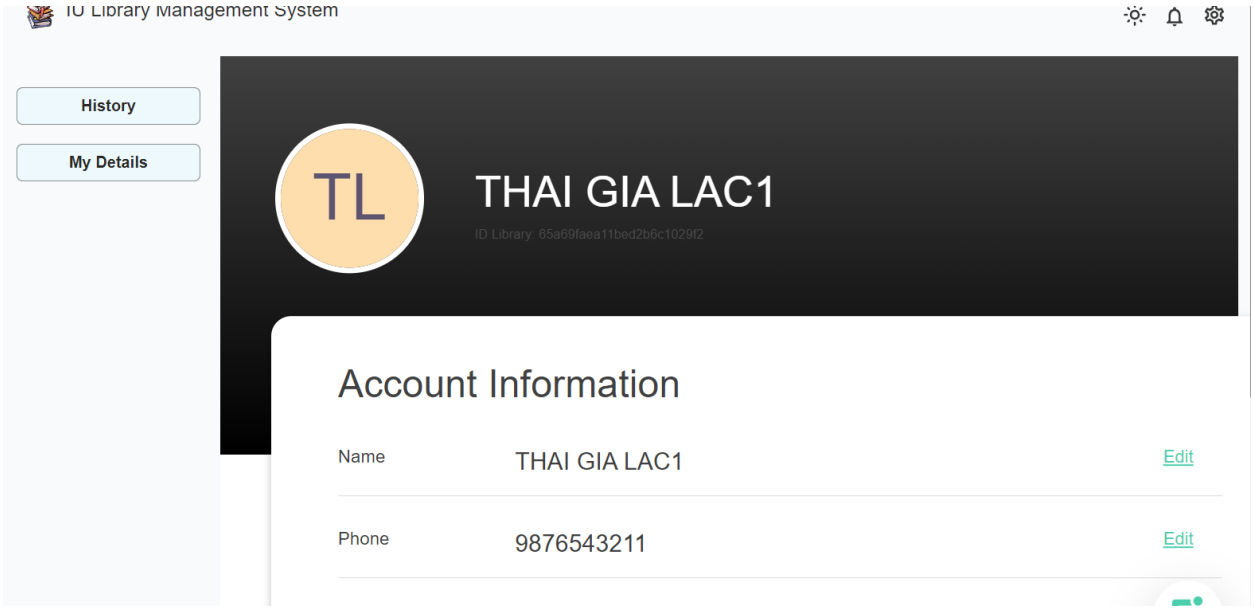


Figure 58: Update username successfully

4.2.10 Admin Home Page:

- After the guest logs in, the account with the admin role will be navigated to the home page for admins. On this home page, the nav bar still has darkmode, notification icon and logout button. The left sidebar shows the Admin role and management services and all the buttons to control the admin dashboard.

- On the admin page, there are greetings at each specified time frame and actual time to help admins manage their working time. The section below includes 6 information boxes about all data in the library database.

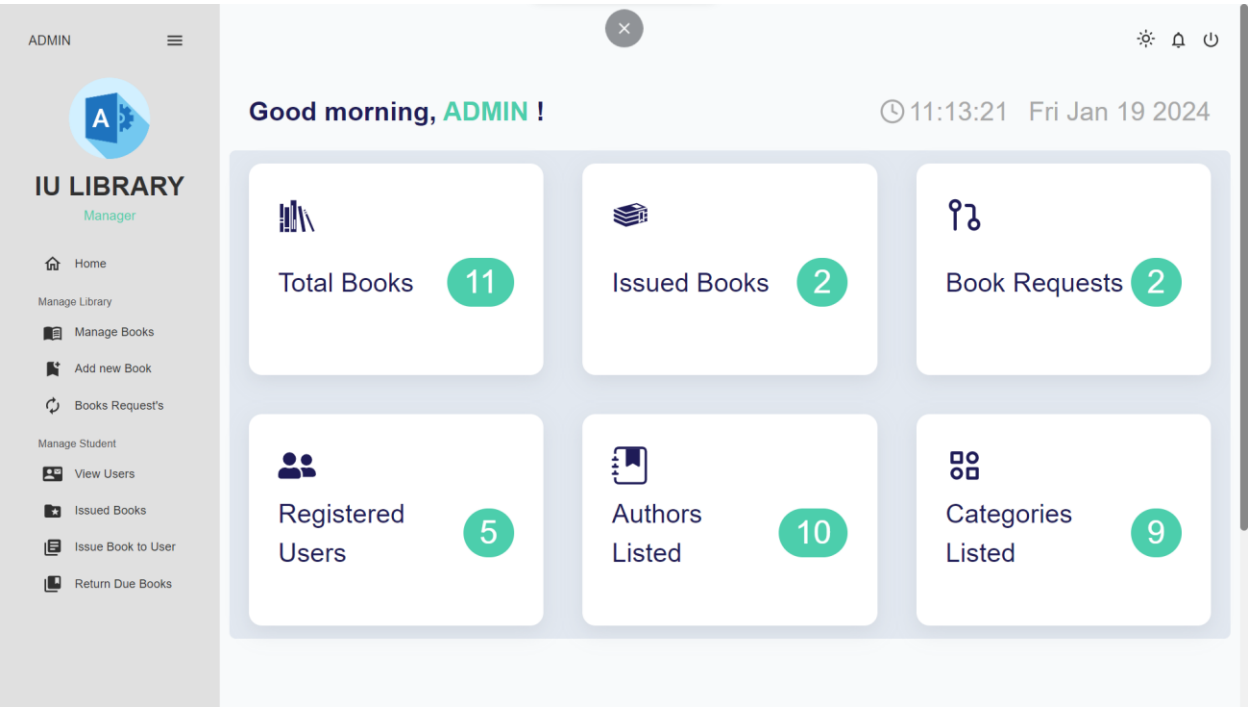


Figure 59: Admin home page

4.2.11 Admin Manage Books Page:

- This is the interface of the admin's book management page. Here, the entire list of created books will be improved. The admin can filter to make finding books easier. When clicking the view details button, the admin will be taken directly to that book information update page.

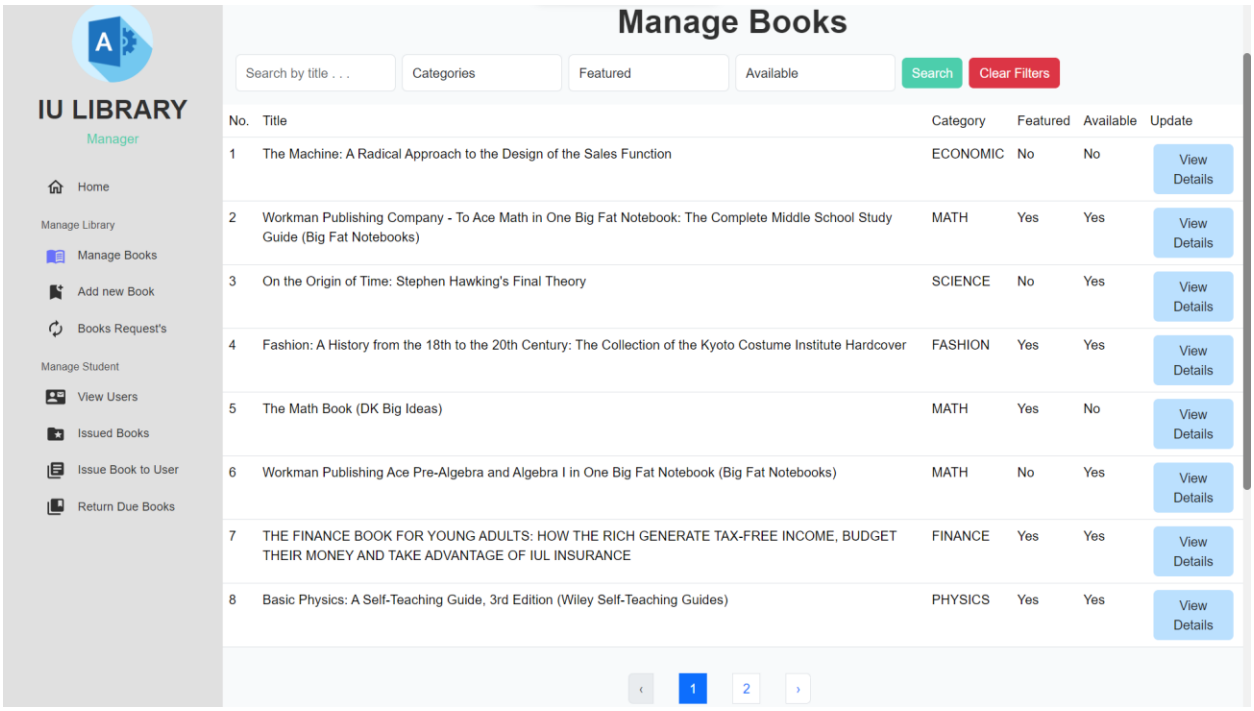


Figure 60: Admin Manage Books Page

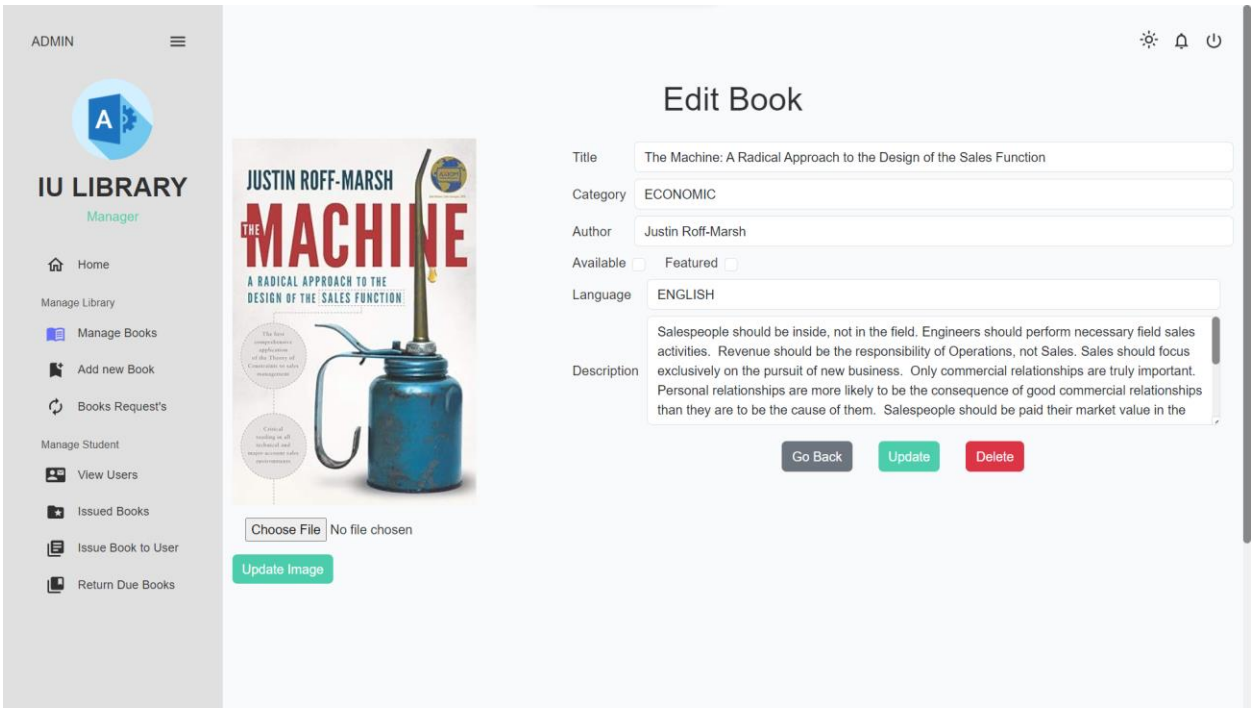


Figure 61: Admin Manage Books Page (Edit Book)

4.2.12 Admin Create Book Page:

- This is the admin's interface for creating a new book. On this page, the admin creates a new book by filling in the book's title, author, description, category, language, availability status, featured and finally uploading the book's cover image file. After submitting, the created book will be displayed in the book management admin page.

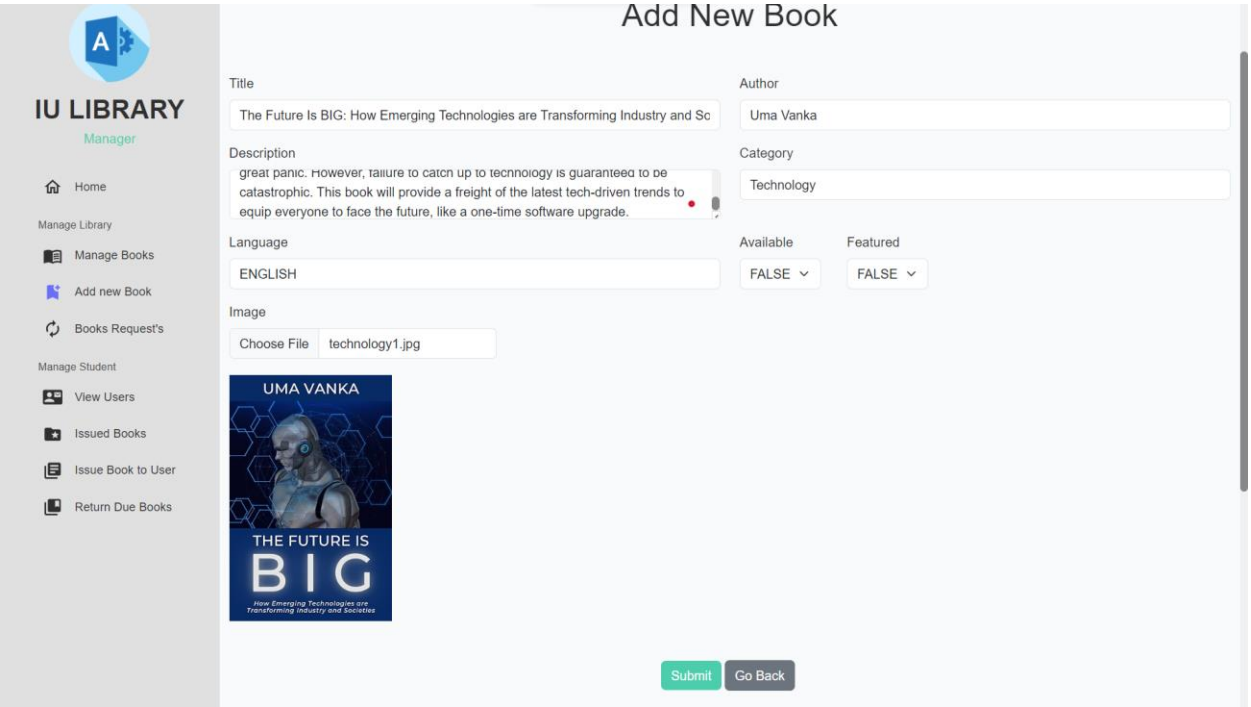


Figure 62: Admin Create Book page.

4.2.13 Admin Books Request's Page:

- This is the interface of the admin's Books Request's page. On this page, the admin can see all book borrowing requests from all students, and the Admin can update the status. The default status of each student when borrowing is "PENDING", the admin can update it to 3 states:

  + "Ready to pick": informs students that books are available at the traditional library, students can go there to pick up books

  + "Accepted": notifies students that the book has been successfully ordered and is waiting for the library to find and send the book to the student in "Ready to pick" status.

+ "Canceled": notifies that the student has failed to borrow a book due to many reasons such as library system rules, the book is out of stock before the student requests to borrow.

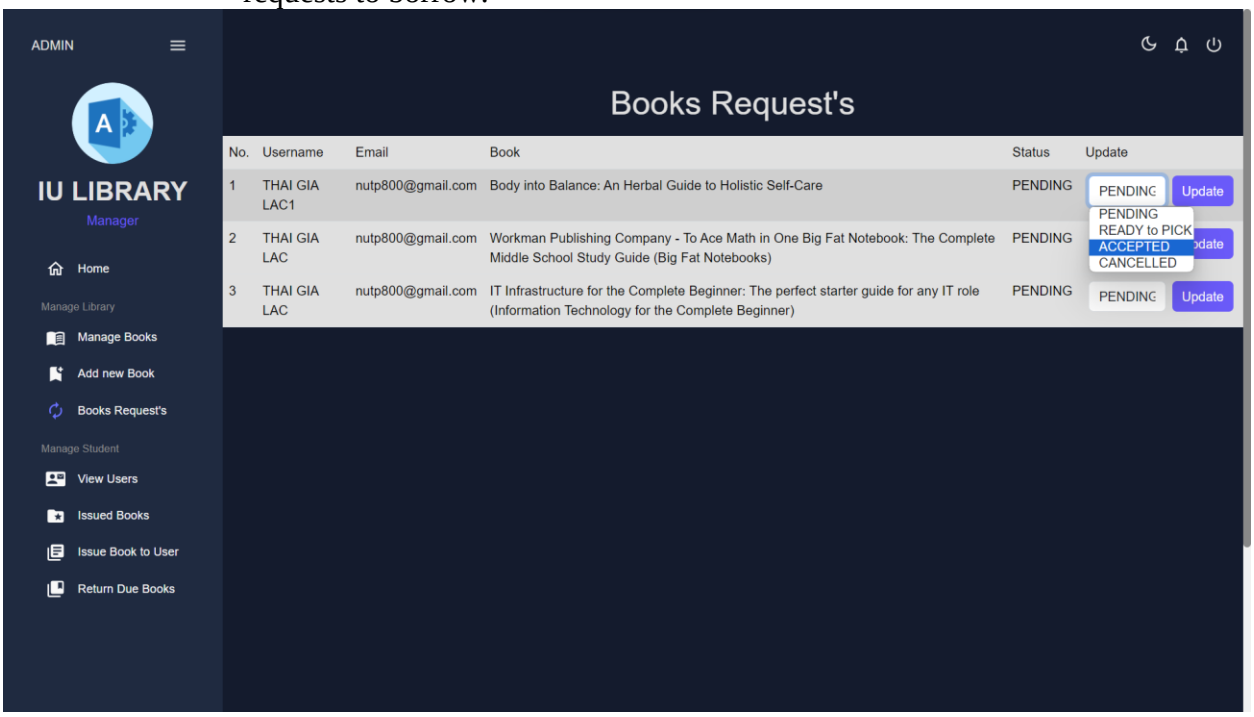


Figure 63: Admin Books Request's page.

4.2.14 Admin View User's Page:

- This is the admin's user view interface. On this page, the admin can view information of all students including name, email, phone, and total books (the number of books students brought home to read and have not returned to the library). school)

- For each student, there is a view detail button next to it. When the admin clicks on it, it will be navigated to the student's information page. On this page, the admin can know which students have or have not authenticated their email accounts and their entire history borrow books.

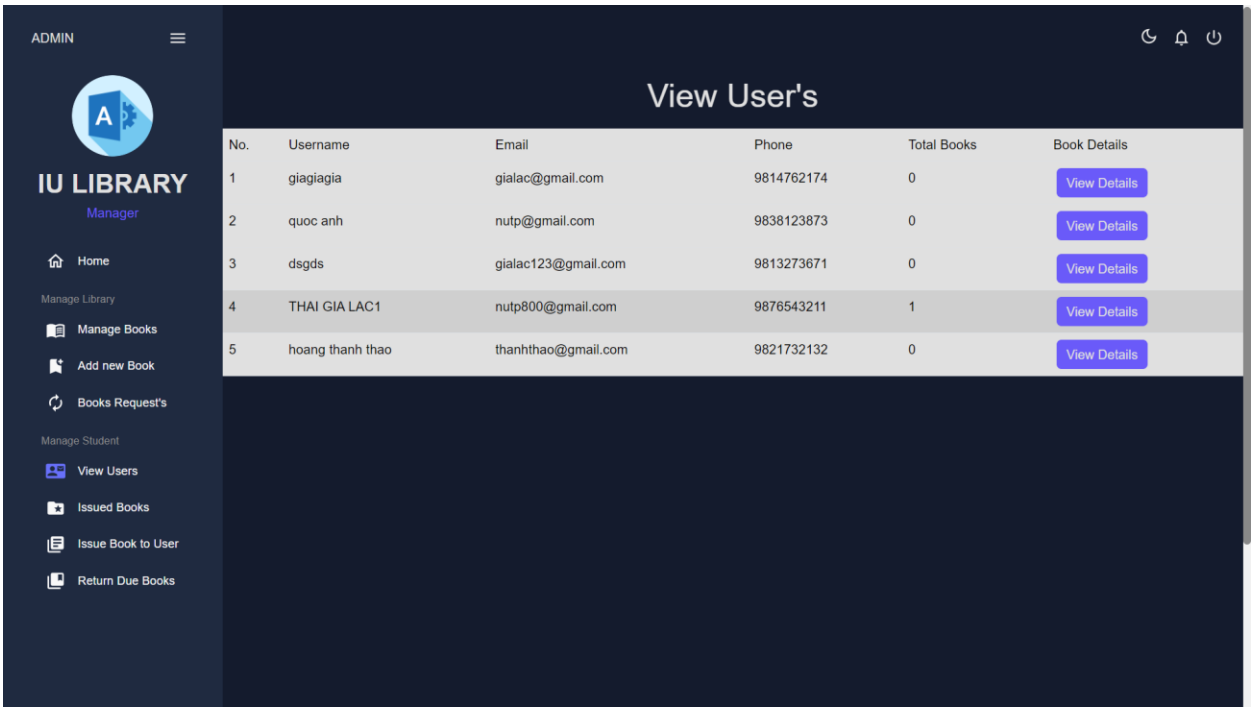


Figure 64: Admin View User's Page

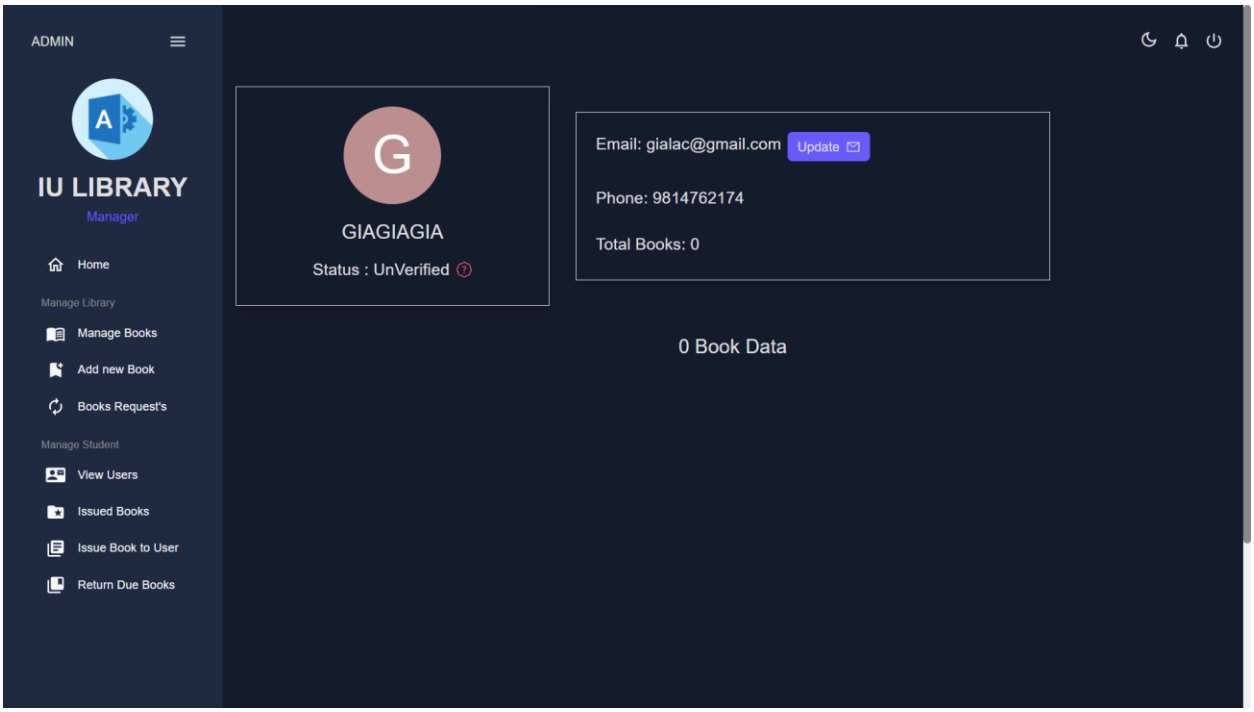


Figure 65: Admin View detail users page (unverified).

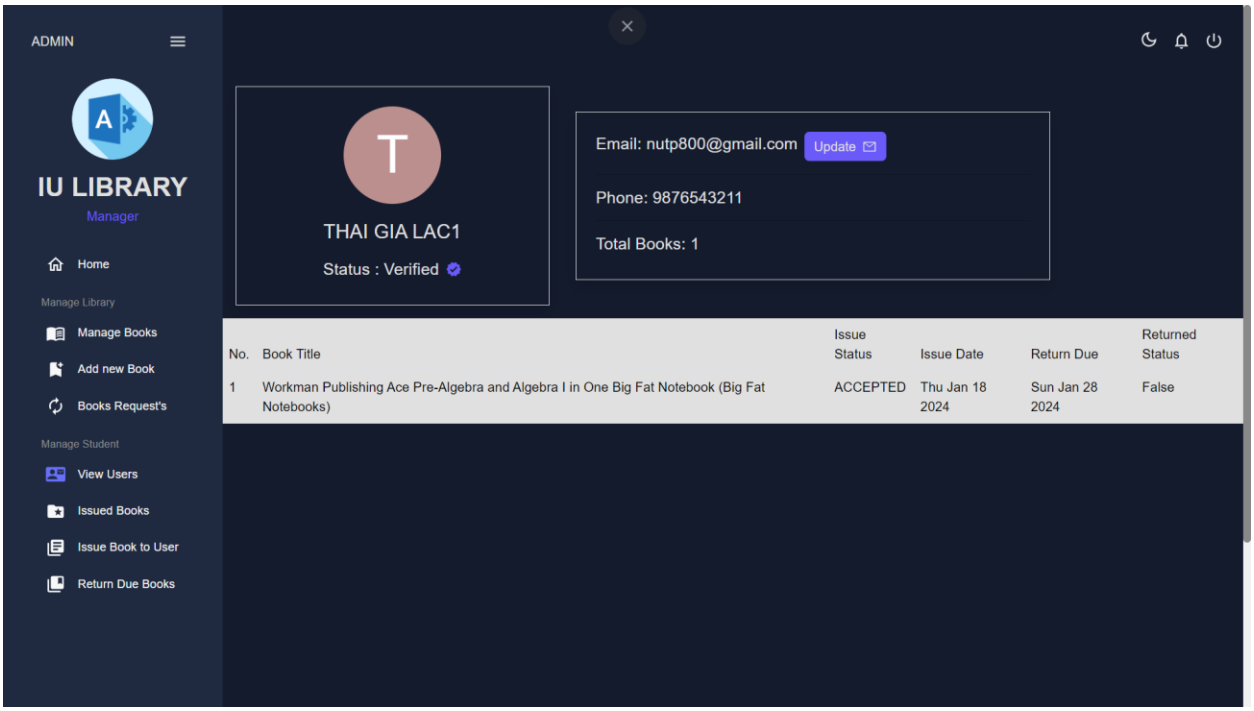


Figure 66: Admin View detail users page (Verified).

4.2.15 Admin Issued Books Page:

- This is the admin's Issued Books page. This page will display all information about students who are in the process of borrowing books but have not yet returned them to the school's traditional library student.

Figure 67: Admin Issued books page

4.2.16 Admin Issue a Book to User Page:

- This is the admin's Issue a Book to User page, on this page the admin can create a book borrowing issue for students to go directly to the traditional library, the admin can get the id of the book the student has borrowed and create a borrowing and return date. Online books on this library website for students.

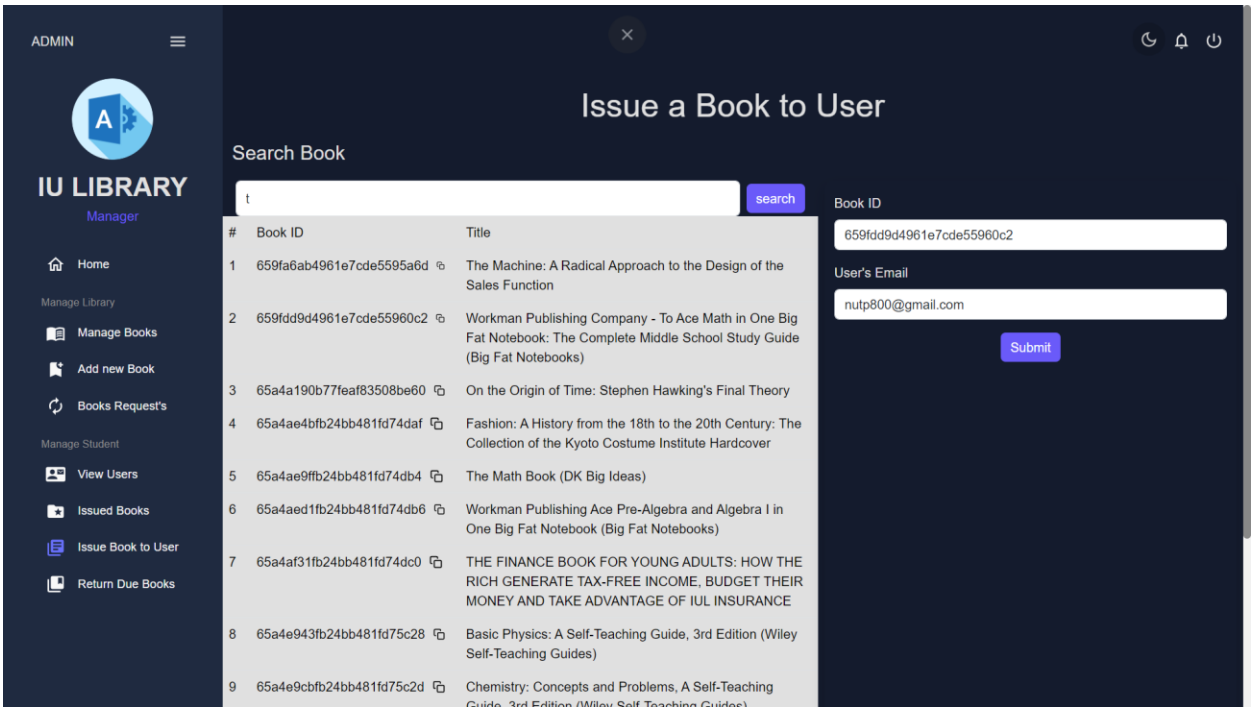


Figure 68: Admin issue a book to user page

4.2.17 Admin Return Due Books Page:

- This is the admin's Return Due Books page. On this page, the admin can confirm that the student has gone to the traditional library to return the book and the "RETURNED" status will be updated in the student's account.

Figure 69: Admin return due books page

CHAPTER 5

DISCUSSION AND EVALUATION

5.1 DISCUSSION

In the realm of digital platforms, the success of every program rests not only on its functioning but also on the user experience it gives. the distinguishing characteristics of an Online Library Management System, stressing its user-friendly interface and optimal performance obtained

A vital component that sets this Online Library Management System apart from others is its dedication to offering consumers with a user-friendly and intuitive interface. The design concept emphasizes simplicity and ease of use, ensuring that students, staff, and administrators can easily move through the system without meeting needless complications.

The user interface is created with an emphasis on clarity and accessibility. Intuitive navigation menus, clear information displays, and strategically positioned action buttons contribute to an environment where users can simply discover the information they need

and complete activities quickly. This attention to user experience corresponds with modern design ideas, creating good encounters and boosting overall happiness.

Definition of user roles and their significance in the system. Explanation of the different user roles in the online library system: administrators, stduent, and guest

Discussion of the specific responsibilities and permissions associated with each user role.

- Improved functionality: each user role has distinct privileges and capabilities, allowing for efficient management of the system.
- Enhanced security: user roles help prevent unauthorized access and ensure data integrity.
- Easy customization: the online library system can be tailored to meet the specific needs and requirements of different user roles.

5.2 EVALUATION

This evaluation explores the user interface, book borrowing processes, and the impact on students and administrators, highlighting the system's user-friendliness and efficiency.

- The user interface (UI) is frequently the initial point of engagement for any system, and in the case of the Online Library Management System, the UI stands out for its remarkable user-friendliness. Both the admin and student dashboards have been deliberately built to be straightforward and easy to browse.
- Furthermore, the introduction of a history function for book borrowing difficulties and requests offers an added degree of ease, enabling students to trace their borrowing history and manage their reading materials effectively.
- Similarly, the admin dashboard mimics this user-friendly design, giving administrators smooth experience in administering the whole library system. The consistency between the admin and student dashboards guarantees a uniform and intuitive experience for all users, leading to a favorable overall engagement with the system.
- The system's capabilities to check the status of borrowed books, manage due dates, and issue timely reminders to students adds to the avoidance of late returns. This not only increases the students' experience but also ensures the methodical management of library resources.
- For administrators, the system proves to be a valuable tool in controlling the complete library infrastructure. The integrated view of the quantity and state of books, together with the capacity to manage student information, simplifies administrative operations. This efficiency eventually translates into a more organized and resourceful library, benefiting both administrators and the student community.

The review of the Online Library Management System emphasizes its achievement in offering a smooth experience for users and managers alike. The user-friendly UI and speedy book borrowing procedures lead to a happy and productive relationship with the system. As students are empowered to borrow and read more books, and administrators effectively manage library resources, the system serves as a monument to the revolutionary potential of technology in the field of education.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The completion of this thesis represents an important milestone in the creation and assessment of the Online Library Management System. Through focused work and cooperation with my excellent adviser, Mrs. Tu, this project has effectively attained its key goals. The user-friendly design, fast book borrowing operations, and influence on both students and administrators illustrate the project's efficiency in strengthening library administration in the digital era.

The focus on a user-centric design philosophy has resulted in dashboards that are not only simple to use but also contribute to a good and intuitive experience for both students and administrators. The history function for book borrowing problems and requests has proved to be a significant addition, supporting proper administration of borrowed materials and giving administrators extensive insights into library activity.

The Online Library Management System project has been a transformative learning experience, enhancing the author's skills in programming languages, research ability, and self-discipline. The project served as an immersive workshop, enhancing the author's proficiency in front-end development and JavaScript, React, and CSS. The author's self-directed learning approach fostered adaptability and a mindset of precision in the face of evolving technological landscapes. The project also required constant concentration, fueled by the intricate details of code implementation and strategic decision-making processes. The project has not only advanced the author's technical proficiency but also cultivated essential life skills, such as effective communication with stakeholders and problem-solving. These skills are invaluable takeaways that will be beneficial for future software development endeavors.

6.2 Future work

While the Online Library Management System has achieved a major point of completion, there remain opportunities for further refinement and growth. The following elements reflect possible opportunities for further work:

- Although the project has made efforts in building a user-friendly interface, maintaining complete responsiveness across all devices is an ongoing aim. Future development will concentrate on enhancing the responsive design to provide great user experiences on multiple screen sizes.
- Enhancing the notification system to contain all important information for both students and administrators is a goal. This might entail integrating real-time alerts for book due dates, system changes, and other vital information to keep users informed and interested.
- Continuous work towards enhancing the system's performance, including fine-tuning database queries, decreasing loading times, and integrating caching methods, will provide a smooth and responsive experience for users.

REFERENCES

[1] GeeksforGeeks. (2023, November 28). Folder structure for a Node JS project. https://www.geeksforgeeks.org/folder-structure-for-a-node-js-project/

[2] Taylor, D. (2023, December 23). *What is MongoDB? Introduction, Architecture, Features & Example*. Guru99. https://www.guru99.com/what-is-mongodb.html

[3] *ExpressJs Tutorial | A comprehensive guide on ExpressJS framework*. (n.d.). Radixweb. https://radixweb.com/introduction-to-expressjs

[4] Express/Node introduction - Learn web development | MDN. (2024, January 1). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[5] Express/Node introduction - Learn web development | MDN. (2024, January 1). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[6] GeeksforGeeks. (2023a, September 12). *React Material UI*. https://www.geeksforgeeks.org/react-material-ui/

[7] Chinda, G. (2023, September 13). *Using Bootstrap with React: Tutorial with examples - LogRocket Blog*. LogRocket Blog. https://blog.logrocket.com/using-bootstrap-react-tutorial-examples/

[8] *Material UI Tutorial | React.school*. (n.d.-b). https://react.school/material-ui

[9] Nym. (2024, January 13). *Library Management System Project Report*. Itsourcecode.com. https://itsourcecode.com/fyp/library-management-system-project-report/

[10] Waheed, A. (2023, December 26). *How to Test an API with Postman*. Apidog Blog. https://apidog.com/blog/how-to-use-postman-for-api-testing/