ARSI UNIVERSITY

COLLEGE OF BUSINESS AND ECONOMICS

DEPARTMENT OF INFORMATION TECHNOLOGY

WEB-BASED ONLINE BOOK STORE SYSTEM FOR ARSI UNIVERSITY

Submitted to Information Technology Department

Done By

No	NAME	ID	EMAIL	PHONE
1.	ZEWUDE DESALEGN	11517/14	desalegnzewude95@gmail.com	0948104278
2.	BINIAM TAFA	13201/14	biniamtafa@gmail.com	0946301189
3.	IFTU BERHANU	11779/14	iftuberhanu@gmail.com	0904209265
4.	DEJACH WORKU	10343/13	dejachworkuit@gmail.com	0995785658
5.	TESFAYE ANSHEBO	11147/14	anshebotesfaye9@gmail.com	0952400978
6.	HANAN KEMAL	13113/14	hanankemal449@gmail.com	0995615887

Advisor: Mr. Regasa A.
Ethiopia, Asella,

Submission date: May /2025

DECLARATION

Date: ----

Group Members:

Full Name	Signature
1. DEJACH WORKU	____________________
2. BINIAM TAFA	______________________
3. IFTU BERHANU	______________________
4. HANAN KEMAL	______________________
5. ZEWUDE DESALEGN	______________________
6. TESFAYE ANSHEBO	______________________

APPROVAL FORM

This is to confirm that the project report Online bookstore for ARU --- **submitted to ARU University, College of Business and Economics, Department of Information Technology** by:Dejach worku, Hanan Kemal, Biniam Tafa , Iftu Berhanu, Zewde Desalegn, Tesfaye Anshebo are approved for submission.

Advisor Name	Signature	Date
--------------------------------	--------------------	--------------
Department Head Name	Signature	Date
--------------------------------	--------------------	-------------
Examiner 1 Name	Signature	Date
--------------------------------	--------------------	-------------
Examiner 2 Name	Signature	Date
--------------------------------	--------------------	-------------
Examiner 3 Name	Signature	Date
--------------------------------	--------------------	-------------

ACKNOWLEDGMENT

We sincerely thank everyone who supported us in completing our project, "Web-Based Online Book Store System for Arsi University." We are truly grateful to our project advisor, Mr. Regassa Alemu, for his constant guidance, helpful advice, and support in writing and documenting this work, which greatly helped us succeed. We also deeply appreciate Mr. Ngusu, the Director of Arsi University Dinsho Campus, for his kind encouragement and for providing the resources we needed to carry out our research and build the system. Additionally, we are thankful to all the librarians at Arsi University for their cooperation and for sharing their knowledge about the current bookstore system, which helped us create a simple and useful online platform that meets the needs of students

Table of Contents

List of Table

List of Figure

List of Acronyms

Admin:	Administrator
ARU:	Arsi university
BR:	Business Rules
CSS:	Cascading Style Sheets
DB:	Database
GUI:	Graphical user interface
HTML:	Hypertext Markup Language
JS:	JavaScript
MS:	Microsoft
MySQL:	My Structured Query Language
PHP:	Hypertext Preprocessor
VS:	Visual Studio
XAMPP:	Cross-Platform Apache MySQL PHP Perl
UAT:	User Acceptance Testing
UC:	use case
UML:	Unified Modeling Language

ABSTRACT

The intention of this project is to develop a supplemental web-based online bookstore system for ARU University. The current ARU bookstore system in library is physical bookstore system manages educational resources manually but this physical bookstore system it has lacks electronic books and digital access. This creates delays in providing timely book availability and burdens students with slow searches. The Web-Based Online Book Store System is an online portal to coordinate book supply and student demand. Its purpose is to bring an online advantage for students, allowing them to search by title, author, and department, read books and download books based on availability introducing e-books to ARU. Rules require user authentication, ban redistribution, and uphold copyright. The system offers details on books, publishers, and authors to improve access for ARU University students.

CHAPTER ONE

INTRODUCTION

In today’s digital time, the way we find and use reading materials has changed dramatically. The Web-Based Online Book Store System is an important step forward in making books and academic resources more accessible for students(Haleem et al., 2022). Traditional bookstores and simple online platforms have been around for a long time, but they often don’t fully meet the needs of students. For example, at ARU University, students face problems like limited book choices, outdated textbooks, and bookstores that are not open when they need them. These challenges make it harder for students to get the materials they need for their studies.

To address these issues, we are developing a Web-Based Online Book Store System designed specifically for ARU University students. This platform will act as a digital bookstore system, offering free access to a wide range of reading materials without requiring purchases. Students will be able to easily browse, search, and read books, download including textbooks, literature, and reference, guide materials, anytime they want. With a simple, user-friendly design and powerful search features, this system will help students quickly find the resources they need.

1.1 BACKGROUND OF THE PROJECT

ARU University, a public university in Ethiopia’s Oromia region, was established in 2006 to advance higher education and contribute to national development. Located in Asella City, it operates three campuses and offers a variety of undergraduate and postgraduate programs in natural and social sciences, focusing on advanced knowledge and research. The university also provides a physical bookstore system to support student access to academic resources. However, this system faces challenges such as outdated textbooks, inefficient search processes, and high operational costs.

1.2 STATEMENT OF THE PROBLEM

The current bookstore system at ARU University operates as a traditional physical store, which creates several challenges for students and staff. While the bookstore provides access to textbooks and reference materials, it lacks the efficiency and convenience of a modern digital system. The existing system does not support online searching, read and downloading of books, making it difficult for students to access resources quickly and efficiently.

So, we have identified the following problems in the existing system:

- Outdated Textbooks: The bookstore often stocks older editions of books, which may not meet the current academic requirements of students.
- Inefficient Search Process: Finding books in a physical store is time-consuming and lacks the convenience of quick online searches. Students often spend significant time locating the books they need, which could be better spent on their studies.
- Limited Accessibility: Students can only access the bookstore during specific operating hours, which may not align with their schedules. This limitation restricts their ability to obtain necessary materials at their convenience.
- High Operational Cost: Running a physical bookstore requires significant manpower for tasks such as stocking shelves, managing inventory, and assisting customers. Additionally, the system generates a large volume of paperwork, which is both resource-intensive and environmentally unfriendly.
- Difficulty in Tracking Demand: The bookstore cannot easily track which books are in high demand and which are rarely requested. This makes it challenging to manage inventory effectively and ensure that students have access to the most needed materials.
- No Real-Time Inventory Updates: The inventory system cannot show the total number of books categorized by department or subject in real-time. This makes it difficult for students to know which books are available before visiting the bookstore.

These issues highlight the need for a more efficient and modern solution to improve access to academic resources for ARU University students. The proposed Web-Based Online Book Store System aims to address these challenges by providing a digital platform for students to easily browse, borrow, and access books anytime and from anywhere.

1.3 OBJECTIVES

1.3.1General Objective of the project

The general objective of this project is to develop an online bookstore system.

1.3.2 Specific Objective of the project.

- Gathering requirements for the online bookstore system.
- Identifying and defining problems in the existing bookstore system.
- Analyzing the current physical bookstore system.
- Identifying functional and non-functional requirements.

- Designing an interactive web-based bookstore interface for users.
- Creating a database to store book records and user profiles.
- Coding and testing the new system.
- Deploying the system for ARU University.

1.4 Feasibility Analysis

To bring the successful completion of this project's goals and objectives, the feasibility issues listed below have determined the project's viability through the discipline of planning, organizing, and managing resources.

1.4.1 Technical feasibility

Technical feasibility considers both the hardware and software requirements. We must determine whether the required technology and proposed equipment have the capacity to store the data used in the project. The system will be developed using web development techniques, as the group members are familiar with the tools and methodologies (e.g., HTML, CSS, JS, PHP, MySQL, and XAMPP server) required to build this system for data collection, coding, and implementation. The team possesses the necessary skills to complete the project, making it technically feasible.

1.4.2 Operational feasibility

The system will provide adequate throughput at the desired time to the users and deliver needed information in a timely and usefully formatted way. It will include security features to grant access privileges by providing accounts for authorized users, such as students and administrators. The system will offer help descriptions to guide users on how to navigate and utilize its features. Any technical modifications or updates to the system will be managed by the developers, ensuring smooth operation and adaptability to user needs.

1.5. Significance and Beneficiary of the Project

1.5.1 Significance of the Project

The current bookstore system at ARU University relies heavily on physical operations, which limits accessibility and efficiency. Developing an online bookstore system will bring significant benefits from multiple perspectives. For students, it provides instant access to a wide range of up-to-date books and resources, eliminating the need for physical visits and saving time. The system offers a user-friendly interface with powerful search capabilities, allowing users to find books by title, author, or department effortlessly. For the university, it reduces operational costs associated with manpower and paperwork, while improving inventory management through real-time updates. This digital platform enhances resource availability, supports academic success, and promotes environmental sustainability by reducing paper usage.

1.5.2 Beneficiaries of the Project

Several groups will benefit from this system. The primary beneficiaries are ARU University students, who will gain easy and convenient access to educational materials anytime and anywhere, enhancing their learning experience. The university itself will benefit from a modernized resource management system that improves efficiency and supports its academic mission. Additionally, the developers will enhance their technical skills, communication abilities, and problem-solving capabilities through the process of designing and implementing this system.

1.6. Scope and Limitation of the project

1.6.1. Scope of Project

- Add administrator.
- Manage administrator.
- Registering Librarian.
- Manage librarian.
- storing books online.
- Managing books (edit, delete, update).
- Registering users.
- Managing user info (edit, delete, update).
- Generate notification.
- Reading and downloading books.
- book ratings.
- Send comment.
- Change password.
- Logout.

1.6.2. Limitation of project

- System cannot support multiple language.

1.7. The methodology of the project

This project involves building a dynamic web-based online bookstore system. To achieve our objectives, an appropriate software design methodology, the agile methodology, has been chosen.

Agile allows for iterative development, breaking the project into smaller phases that build upon each other. We selected this approach because it enables flexibility, continuous user feedback, and early testing of ideas, reducing time spent on documentation and focusing more on design and implementation.

1.7.1. Data collection tools and techniques

To fulfill the project's purpose, we gathered extensive information using various methods to understand the current bookstore system and its challenges. Primary data sources include interviews with students and library staff to capture their needs and experiences, and observation of the existing bookstore workflow to identify inefficiencies. Secondary data sources, such as online resources and tutorials from the internet, provide additional insights into available samples and development techniques, supporting the system's design and implementation.

1.7.2. System Analysis and Design

For system analysis and design, we employed specific tools and techniques. The Unified Modeling Language (UML) is used as the object-oriented modeling language to create diagrams such as use cases, class diagrams, and sequence diagrams, facilitating analysis and design phases. Tools like Draw.io and Microsoft Visio assist in visualizing these models.

For interface design, we utilize VS Code editor to craft a responsive and user-friendly layout with HTML, CSS, and JS, ensuring the system meets user expectations and technical requirements effectively.

1.7.3 System Development Models

The development method chosen for this Online Book Store System is the agile methodology. Agile is an iterative and incremental approach that emphasizes flexibility, collaboration, and continuous improvement throughout the development process. This model is ideal for our project as it allows us to break the development cycle into smaller, manageable phases, enabling us to adapt to changing requirements and incorporate user feedback effectively

The Agile process involves repeated cycles of.

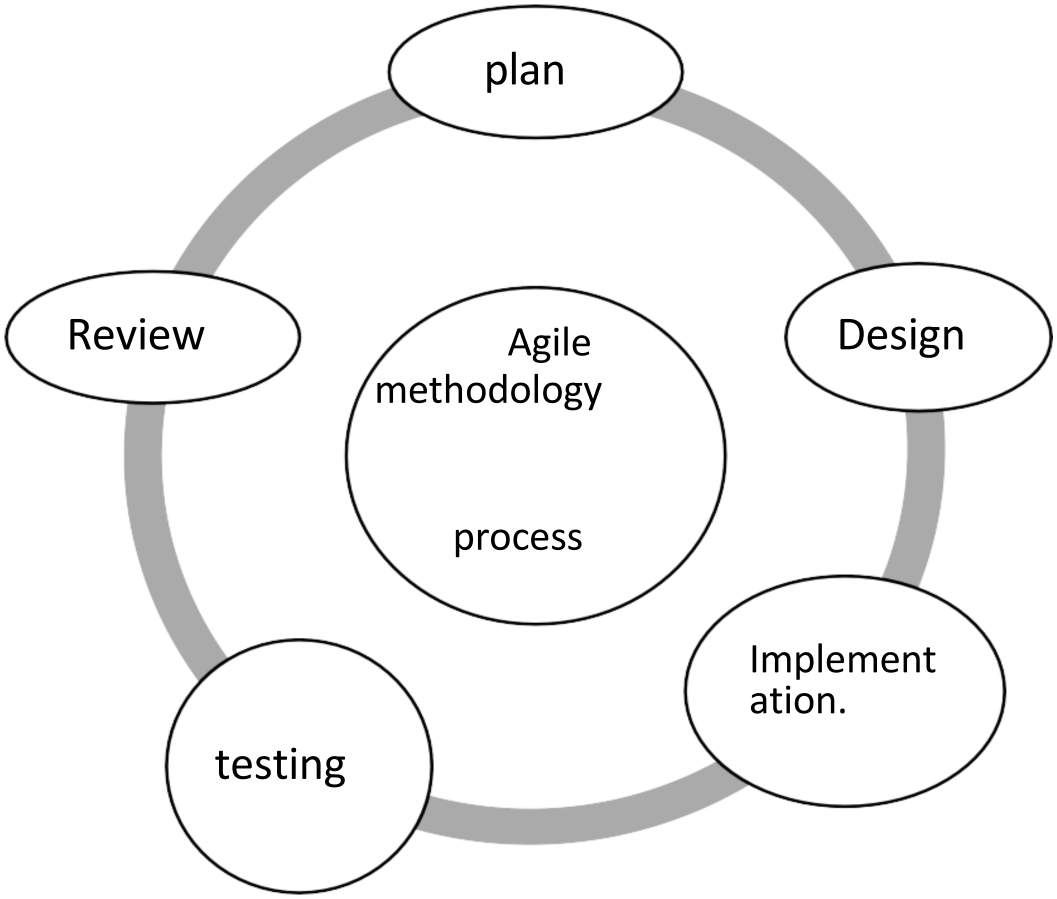


Figure 1:1iterative model.

1.7.4. System Testing Methodology

To ensure the "Web-Based Online Book Store System for ARU University" meets its functional and non-functional requirements, we will adopt a comprehensive testing methodology across multiple levels

- **Unit Testing**: Tests individual parts (e.g., login, search) alone using PHP Unit to ensure they work right, like checking if search finds the correct book.
- **Integration Testing**: Checks if parts work together smoothly by testing user actions, catching connection issues early.
- **System Testing**: Tests the whole system with real scenarios.
- **UAT**: Real users (students, Library staff) test the system to see if it's easy and useful.

1.8. System Development Tools and Techniques

The team plan to use the following software Development tools for the development of the system among different available tools.

Activities	**Tools**
Documentation	Microsoft Word 2010
Code Editing	Visual Studio
Database Server	MySQL (via XAMPP)
Web Server	Apache (via XAMPP)
Client-Side scripting	JavaScript
Back-end Implementation	PHP
Interface Design	HTML, CSS
Diagramming	Draw.io, MS Visio
Presentations	Microsoft PowerPoint

Table 1:System Development

CHAPTER TWO

2. DESCRIPTION OF EXISTING SYSTEM

2.1. Introduction to Existing System

The current bookstore system at ARSI University operates as a traditional physical bookstore located on campus, managed by university library staff to provide students and college with access to academic resources such as textbooks, reference books, and supplementary materials. This system relies heavily on manual processes for inventory management, book lending, and record-keeping, with no digital platform for online access or transactions. While it serves the basic purpose of distributing educational materials, it lacks modern features like online browsing real- time inventory updates, which limits its efficiency and convenience for users.

2.2. Users of Existing System

The existing bookstore system at ARU University involves several key users who perform distinct activities to manage and access resources.

1. **Library Director**: The Director oversees the entire library operation, ensuring that all services run smoothly and efficiently. They set policies, manage budgets, and make high-level decisions to maintain the system's effectiveness.
2. **Group Leader**: The Group Leader manages a specific team within the library. They coordinate activities, ensure that tasks are completed efficiently, and serve as a liaison between the library staff and the Director, keeping workflows aligned with university goals.
3. **Catalog Arranger**: Catalog Arrangers are responsible for organizing and maintaining the library's collection. They categorize books by subject or department, update the catalog system, and ensure that materials are easily accessible to users on the shelves.
4. **Book Borrower**: Book Borrowers manage the lending process. They check books in and out, track borrowing records using manual logs or basic systems, and ensure that all materials are returned on time, issuing reminders for overdue items.
5. **Acquisition Officer**: The Acquisition Officer handles the procurement of new books and resources. They work with publishers, review faculty requests, and ensure the library's collection stays current and relevant to academic needs.

6. **Reference Librarian**: Reference Librarians assist students and faculty with research. They guide users to specific books or journals, provide expertise on academic resources, and help navigate the library's limited catalog for complex inquiries.
7. **Maintenance Staff**: Maintenance Staff keep the physical space functional. They shelve returned books, repair damaged materials, and ensure the library remains clean and organized for users to access resources comfortably.
8. **Users:**

- Visiting the bookstore during operating hours.
- Browsing available titles on shelves.
- Selecting required textbooks for studies.

2.3. Major Function of Existing Systems

The existing bookstore system at ARSI University is a physical operation designed to provide students and collage with access to academic books for educational purposes. It relies on manual processes managed by Library staff to support borrowing and resource availability. The major functions of the current system are:

- **Book Acquisition**: Library Staff obtain books from university suppliers or publishers based on course requirements provided by collage, ensuring relevant materials are available for the reader and borrower.
- **Inventory Management**: The system tracks the number of books in stock using paper records or basic digital tools like spreadsheets.
- **Book Lending**: Students and collage members (instructor, other collage users) read, borrow books by visiting the bookstore, where staff record borrower details and due dates manually on forms.
- **User Support**: Library Staff assist users in finding books by answering inquiries and directing them to available resources.

2.4. Forms and Other Documents of the Existing Systems

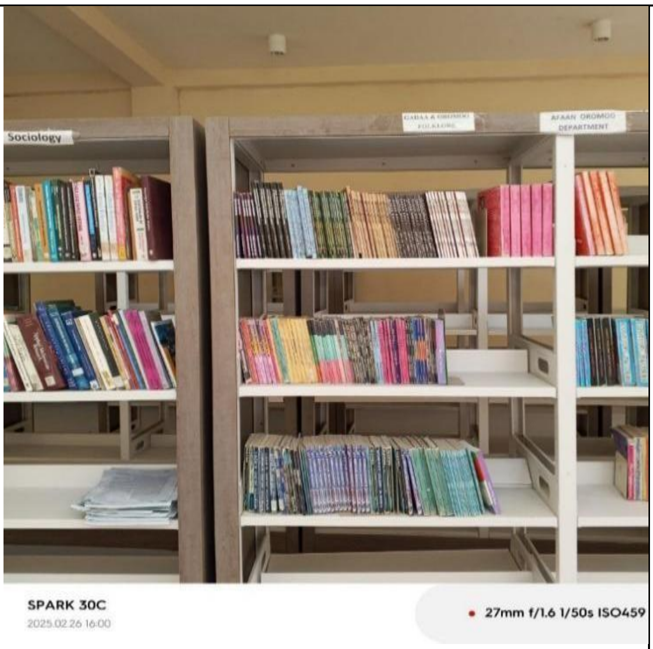


Figure 2:exist book shelves system.

Figure 3:exist book search system

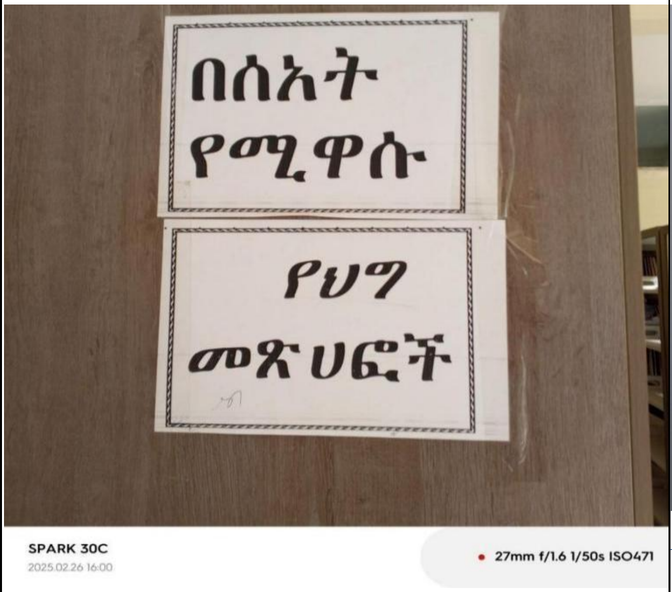
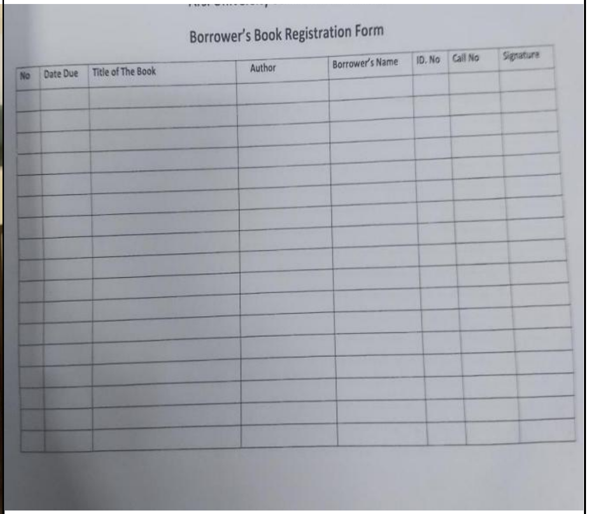


Figure 4: exist book limitation



Borrower's Book Registration Form

No	Date Due	Title of The Book	Author	Borrower's Name	ID. No	Call No	Signature

Figure 5: exist book borrow system

2.5. Drawbacks of the Existing System

The current physical bookstore system at ARU University has several issues that make it less helpful for students. Many books on the shelves are old and don't match what students need for their current classes. Finding a book is slow because students have to search through shelves by hand, and it's frustrating when the book they want isn't there. Even though the bookstore is open 24 hours, it's still hard to use because there is no online way to check what's available, so students must visit in person every time. Library Staff have to manage everything manually, like keeping track of books and recording who borrows them, which takes a lot of work and can lead to mistakes.

2.6. Business rule of the existing system

Business rules define the policies and procedures that govern the current physical bookstore system at ARSI University, ensuring students and college users can access academic materials for reading and borrowing. These rules maintain an organized approach to resource sharing within the university community. Below are the key business rules:

Name	Access Eligibility
ID	BR1
Description	Only registered ARSI University students and collage Employer can read or borrow books from the bookstore.

Table 2: Business Rule 1

Name	Borrowing Limit
ID	BR 2
Description	Each user can borrow up to 5 books based on the book availability books at a time to ensure fair access to borrowing while allowing others to read in-store.

Table 3: Business Rule 2

Name	Return Deadline
ID	BR 3
Description	Borrowed books must be returned within a given days (1week - 1 semester), while reading in the bookstore has time limit.

Table 4: Business Rule 3

Name	Inventory Update
ID	BR 4
Description	Library Staff must manually update the inventory record after every borrowing or return, and note books reserved for in-store reading

Table 5: Business Rule 4

These rules ensure that the bookstore supports both borrowing books to take away and reading them on-site, balancing availability and access.

CHAPTER THREE

3. PROPOSED SYSTEM

3.1. System Description

The Web-Based Online Book Store System for ARU University is a web-based platform designed to modernize the current physical bookstore by enabling students and college to access academic books digitally copies from anywhere with an internet connection. This system allows users to search for books, read them online digital copies, and download them to their devices, offering a flexible and convenient alternative to the traditional setup. It aims to streamline how the university community interacts with educational resources like textbooks, reference books, and study materials.

System Features

- ❖ **User Management**: Students and Collage user register and log in with secure accounts to use the system.
- ❖ **Book Browsing and Reading**: Users can search books by title, author, or department and read them directly on the platform.
- ❖ **Downloading**: Users can download books to their devices for offline access with no limitation on the number, making resources fully available anytime.
- ❖ **Notifications**: Automated announce new books add on the system.

Benefits

- ❖ **Anytime Access**: Users can read or download books 24/7, not limited by physical bookstore hours.
- ❖ **Convenience**: Fast searches and the option to download books save time and effort for studying.
- ❖ **Wider Reach**: Digital copies mean more users can access books at once, whether reading online or downloading.
- ❖ **Simplified Management**: Staff can update the digital collection easily, reducing manual record-keeping.

3.2. Functional Requirement of Proposed System

Functional requirements define what the System must do to meet the needs of its users. The following are the key functionalities:

- User Registration and Login: The system must allow students and college employer to create accounts with unique usernames and passwords, and log in to access features like reading and downloading books.
- Search for Books: Users can search the database for books by title, author, or department, retrieving a list of matching results to read or download.
- Read Books Online: The system must enable users to view and read book content directly on the platform without needing to download them.
- Download Books: The system must allow users to download books as digital files (e.g., PDFs) to their devices for offline access, ensuring flexibility in studying.
- Manage Book Inventory: administrator and librarian can add new books, update existing records and
- Generate Notifications: The system sends automated notification when the system adds new books.

- View Book Details: Users can access information about each book, such as author, department, and availability status, to decide whether to read or download it.

3.2. Non-functional Requirements

The non-functional requirements of the "Web-Based Online Book Store System for Arsi University" describe how well the system delivers its services to users, ensuring it is efficient, secure, and easy to use for students, college, and staff accessing academic books.

3.2.1. User Interface and Human Factors

The system will feature a clean and simple web interface, allowing users to quickly search, read or download books without confusion. It must be easy to navigate, with clear buttons and labels, so even users with little computer experience can use it comfortably.

3.2.2. Hardware Consideration

The system requires a strong server with high memory and fast processing to support many users reading or downloading books at once. On the user side, it will work on common devices like laptops, tablets, or phones with a stable internet connection.

3.2.3. Security Issues

The system must protect user data, like login details, with secure encryption and limit access based on roles: staff can manage books, while students and college users can only read or download. This keeps the system safe from unauthorized changes or data leaks.

3.2.4. Performance Consideration

The system should load book searches and pages fast, within a few seconds, even when many users are online during busy times like exams. It must handle downloading and reading smoothly to keep users from waiting too long.

3.2.5. Error Handling and Validation

If a user makes a mistake, like typing a wrong password the system will show a clear error message and let them try again. It will check inputs to stop errors, ensuring actions like downloading work correctly.

3.2.6. Quality Issues

The system must stay online and work well almost all the time, so users can rely on it for reading or download books whenever they need. It should also adjust to different screen sizes and browsers for a consistent experience.

3.2.7. Backup and Recovery

All book files and user data will be backed up every day to a safe place, so if something goes wrong, like a server crash, the system can be fixed quickly with no data lost. Administrator can restore everything within a short time.

3.2.8. Physical Environment

The server will be kept in a locked, cool room at the university to protect it from damage or theft. Users can access the system from anywhere classrooms, dorms, or home as long as they have internet.

3.2.9. Resource Issues

The system must support many users at once, like 500 students downloading or reading during peak times, without slowing down. It should also work well on the university’s network, even if the internet isn’t super-fast.

3.2.10. Documentation

The system will come with easy guides for users on how to read or download books, and for staff on how to add or update books. This helps everyone use it properly and supports future improvements by developers.

CHAPTER FOUR

4. SYSTEM ANALYSIS

4.1. System Model

This section describes the key scenarios for the Web-Based Online Book Store System at Arsi University, focusing on interactions between the three main actors Admin, Librarian, and User (Student) and the system.

Main Actors

1. **Admin**,
2. **Librarian** and
3. **User (Student).**

Scenario Name: Login

First, the actor (Admin, Librarian, or User) visits the system's login page. The actor enters their username and password in the fields. Upon clicking "Login," the system checks the credentials. If valid for a User, it opens the user dashboard. If valid for a Librarian, it loads the librarian dashboard. If valid for an Admin, it directs to the admin dashboard. If invalid, an error message appears, urging re-entry.

Scenario Name: Add Administrator

First, the Admin visits the system's login page and logs in. The admin navigates to the administrator management section on the admin dashboard. The admin selects the option to add a new administrator. The system displays a form requiring details like name, email, username, and password. The admin fills in the fields and clicks "Submit." The system verifies the information. If valid, it creates the new administrator account and shows a confirmation message. If invalid, an error message appears, prompting corrections.

Scenario Name: Manage Administrator

First, the Admin visits the system’s login page and logs in. The admin navigates to the administrator management section. The system lists all existing administrators. The admin selects an administrator to edit or delete. For editing, the system shows the administrator’s details, and the Admin updates fields like email or name, then clicks "Save." For deletion, the admin confirms the action. The system processes the request. If successful, it updates the administrator list and displays a confirmation. If unsuccessful, an error message appears, urging retry.

Scenario Name: Register Librarian

First, the Admin visits the system’s login page and logs in. The Admin navigates to the librarian management section on the admin dashboard. The Admin chooses to add a new librarian. The system presents a form for details such as name, email, username, and password. The Admin completes the form and clicks "Submit." The system checks the input. If valid, it creates the librarian account and displays a success message. If invalid, an error message appears, requesting corrections.

Scenario Name: Manage Librarian

First, the Admin visits the system’s login page and logs in. The Admin navigates to the librarian management section. The system displays a list of librarians. The Admin selects a librarian to edit or delete. For editing, the system shows the librarian’s details, and the Admin updates fields, then clicks "Save." For deletion, the Admin confirms the action. The system processes the request. If successful, it updates the librarian list and shows a confirmation. If unsuccessful, an error message appears, prompting retry.

Scenario Name: Register User

First, the actor (Admin or Librarian) visits the system’s login page and logs in. The actor navigates to the user registration section on their dashboard. The system displays a form for user details like name, email, username, and password. The actor fills in the fields and clicks "Submit." The system verifies the data. If valid, it creates the new user account and shows a confirmation message. If invalid, an error message appears, urging re-entry.

Scenario Name: Manage User Info

First, the actor (Admin or Librarian) visits the system’s login page and logs in. The actor navigates to the user management section. The system lists all registered users. The actor selects a user to edit or delete. For editing, the system displays the user’s details, and the actor updates fields like email, then clicks "Save." For deletion, the actor confirms the action. The system processes the request. If successful, it updates the user list and displays a confirmation. If unsuccessful, an error message appears, prompting retry.

Scenario Name: Store Books Online

First, the actor (Admin or Librarian) visits the system’s login page and logs in. The actor navigates to the book management section and selects the option to add a book. The system presents a form for book details like title, author, and department. The actor completes the form and clicks "Submit." The system checks the input. If valid, it adds the book to the inventory and displays a success message, triggering a notification to users. If invalid, an error message appears, requesting corrections.

Scenario Name: Manage Books

First, the actor (Admin or Librarian) visits the system’s login page and logs in. The actor navigates to the book management section. The system lists all books in the inventory. The actor selects a book to edit or delete. For editing, the system shows the book’s details, and the actor updates fields like title or author, then clicks "Save." For deletion, the actor confirms the action. The system processes the request. If successful, it updates the inventory and shows a confirmation. If unsuccessful, an error message appears, urging retry.

Scenario Name: Send Notification for New Book

First, the actor (Admin or Librarian) visits the system’s login page and logs in. The actor adds a new book through the book management section. The system automatically generates a notification with details like the book’s title and author. The system sends the notification to all users. Upon users’ login, the notification appears on their dashboard. If the notification fails to send, an error message logs the issue for review.

Scenario Name: Approve Comment

First, the actor (Admin or Librarian) visits the system’s login page and logs in. The actor navigates to the comment management section. The system lists all pending user comments. The actor selects a comment to review. The actor chooses to approve.

Scenario Name: Change Password

First, the actor (Admin, Librarian, or User) visits the system’s login page and logs in. The actor navigates to the profile settings section. The system displays a form for changing the password. The actor enters their current password, new password, and confirmation, then clicks "Submit." The system verifies the input. If valid, it updates the password and shows a success message. If invalid, an error message appears, urging re-entry.

Scenario Name: Read and Download Books

First, the actor (Admin or User) visits the system’s login page and logs in. The actor navigates to the book catalog or search section. The system displays available books. The actor selects a book. For reading, the actor clicks "Read Online," and the system loads the book in the browser. For downloading, the actor clicks "Download," and the system prompts a file save. The actor saves the file. If successful, the system confirms the action. If the book is unavailable, an error message appears, suggesting another selection.

Scenario Name: Book Reviews and Ratings

First, the User visits the system’s login page and logs in. The User navigates to a book’s details page. The system displays options to rate. The User selects a rating click "Submit." The system checks the input. If valid, it saves the review and rating, pending approval, and shows a confirmation.

Scenario Name: Send Comment

First, the User visits the system’s login page and logs in. The User navigates to a book’s details page. The system shows a comment submission form. The User enters a comment and clicks "Submit." The system verifies the input. If valid, it saves the comment, pending approval, and displays a confirmation. If invalid, an error message appears, prompting re-entry.

Scenario Name: Logout

First, the actor (Admin, Librarian, or User) visits the system's login page and logs in. The actor navigates to their dashboard and locates the "Logout" option. The actor clicks "Logout." The system ends the session, clears the actor's data, and redirects to the login page. If successful, a confirmation message appears. If an error occurs, an error message prompts retry.

4.1.1 Use Case Model

The second step in system analysis is to build the use case model, which visually depicts the Web-Based Online Book Store System's interactions with its external environment. The main actors are Admin, Librarian, and User (Student), each with distinct roles. All actors must log in to access the system, ensuring secure interactions. Users read, download, review, rate, and comment on books, while Librarians manage books and users, send notifications, and moderate comments. Admins oversee administrators, librarians, users, books, notifications, and comments, with access to reading and downloading. The use cases below outline these interactions, enabling a modern digital platform for AR University's bookstore

4.1.1.2 Proposed System Use Case Diagram

The proposed system use case diagram provides a detailed representation of the Web-Based Online Book Store System, incorporating all major functionalities and interactions between actors and the system.

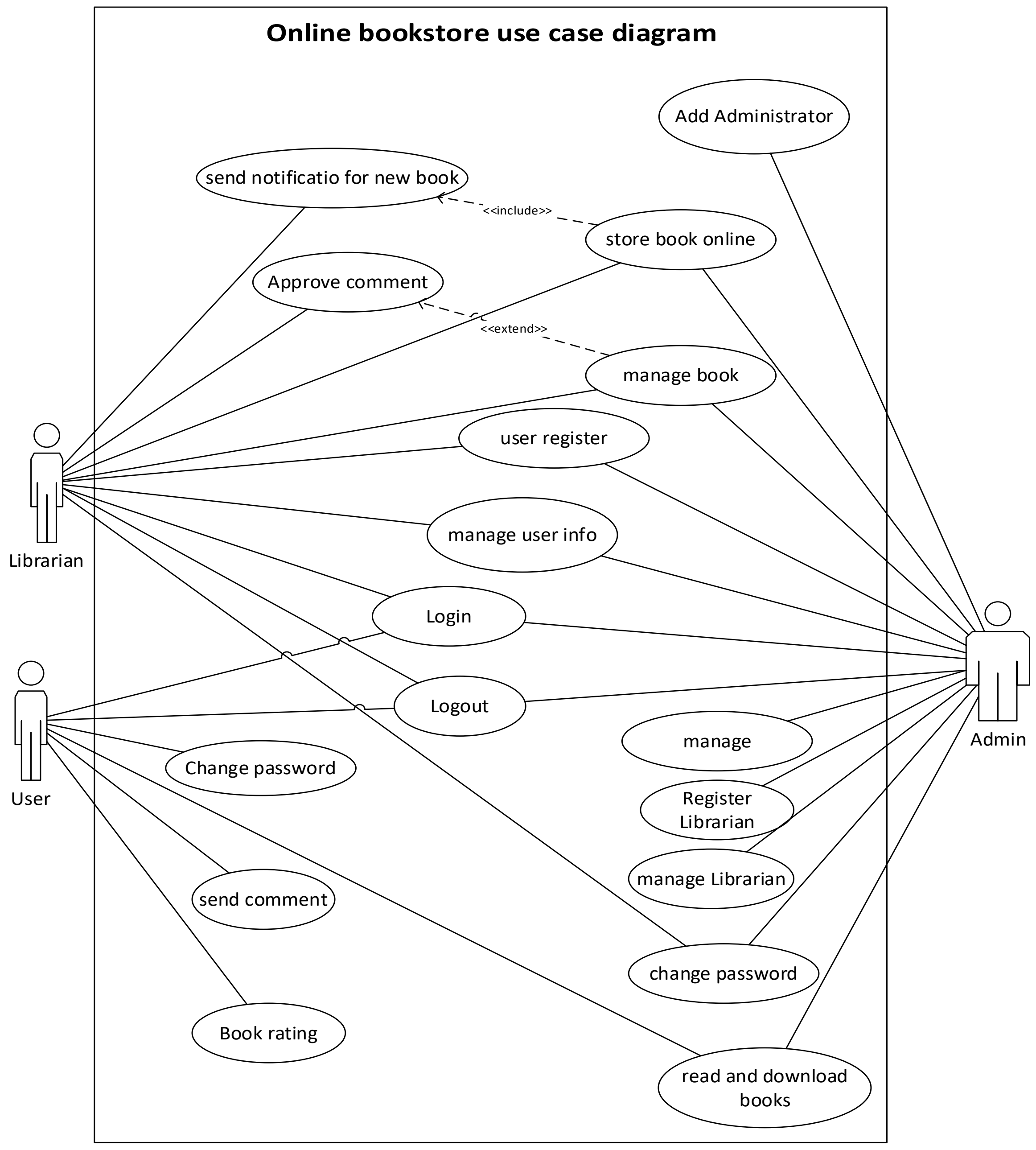


Figure 6: use case diagram

4.1.1.3 Use Case Documentation

Use Case ID	UC#1	
Use case name	login	
Actors	Admin, Librarian, User (Student)	
Description	The actor authenticates their identity by entering a valid username and password to access the system's features based on their role	
Goal	To allow authorized actors to securely access their respective dashboards (Admin, Librarian, or User) for system interaction.	
Precondition	The actor must have a registered account with valid credentials. The system must be accessible via a web browser with an active internet connection	
Basic flow action	actor	System response
	1. The actor navigates to the system's login page. 2. The actor enters their username and password in the provided fields. 3. The actor clicks the "Login" button.	4. The system validates the credentials against the database. 5. If credentials are valid, the system redirects the actor to their role-specific dashboard (Admin: admin dashboard, Librarian: librarian dashboard, User: user dashboard). 6. The system displays a welcome message on the dashboard.
Postcondition	The actor is logged into the system and can access features corresponding to their role. The session is active until logout or timeout	
Alternative Actions	- If the username or password is incorrect, the system displays an error message and prompts the actor to re-enter the credentials. - If the actor forgets their password, the system displays a "Forgot Password" option and prompts the actor to enter their registered email for a reset link (out of scope for this use case)	

Table 6: Use Case Documentation for login

Use Case ID	UC#2	
Use case name	Add Administrator	
Actors	Admin	
Description	The Admin creates a new administrator account to delegate system management tasks.	
Goal	To successfully add a new administrator account to the system, enabling additional administrative access.	
Precondition	The Admin must be logged into the system with valid credentials and have access to the administrator management section.	
Basic flow action	actor	System response
	1. The Admin navigates to the administrator management section on the admin dashboard. 2. The Admin selects the "Add Administrator" option. 3. The system displays a form requiring details such as name, email, username, and password. 4. The Admin fills in the form and clicks "Submit."	5. The system validates the input for uniqueness (username, email) and format. 6. If valid, the system creates the new administrator account in the database. 7. The system displays a confirmation message: "Administrator added successfully.
Postcondition	A new administrator account is created and stored in the database, ready for use.	
Alternative Actions	- If the input data is invalid (e.g., duplicate username or missing fields), the system displays an error message and prompts the Admin to correct the information. - If the email is already registered, the system displays an error message and prompts the Admin to use a different email. - If the database fails to save the account, the system displays an error message and prompts the Admin to try again	

Table 7: Use Case Documentation for add administrator

Use Case ID	UC#3	
Use case name	Manage Administrator	
Actors	Admin	
Description	The Admin edits or deletes existing administrator accounts to maintain system access control.	
Goal	To update or remove an administrator account to reflect current system management needs.	
Precondition	The Admin must be logged into the system and have access to the administrator management section. At least one administrator account must exist.	
Basic flow action	actor	System response
	1. The Admin navigates to the administrator management section. 2. The system displays a list of existing administrators. 3. The Admin selects an administrator to edit or delete.	Edit Path: - The system displays a form with the administrator's details. - The Admin updates fields (e.g., email, name) and clicks "Save." - The system validates the input and updates the database. - The system displays a confirmation message: "Administrator updated successfully." Delete Path: - The Admin confirms the deletion action. - The system removes the account from the database. - The system displays a confirmation message: "Administrator deleted successfully."
Postcondition	The selected administrator account is either updated with new details or deleted from the database.	
Alternative Actions	- If the updated input data is invalid (e.g., invalid email format or missing fields), the system displays an error message and prompts the Admin to correct the information. - If the Admin attempts to delete their own account, the system displays an error message and prompts the Admin to select a different account.	

Table 8: Use Case Documentation for manage administrator

Use Case ID	UC#4	
Use case name	Register Librarian	
Actors	Admin	
Description	The Admin creates a new librarian account to manage books and users in the system.	
Goal	To successfully add a new librarian account, enabling book and user management capabilities.	
Precondition	The Admin must be logged into the system and have access to the librarian management section.	
Basic flow action	actor	System response
	1. The Admin navigates to the librarian management section on the admin dashboard. 2. The Admin selects the "Add Librarian" option. 3. The system displays a form requiring details such as name, email, username, and password. 4. The Admin fills in the form and clicks "Submit."	5. The system validates the input for uniqueness and format. 6. If valid, the system creates the new librarian account in the database. 7. The system displays a confirmation message: "Librarian added successfully."
Postcondition	A new librarian account is created and stored in the database, ready for use	
Alternative Actions	- If the input data is invalid (e.g., duplicate username or missing fields), the system displays an error message and prompts the Admin to correct the information. - If the email is already registered, the system displays an error message and prompts the Admin to use a different email. - If the database fails to save the account, the system displays an error message and prompts the Admin to try again	

Table 9: Use Case Documentation for add librarian

Use Case ID	UC#5	
Use case name	Manage Librarian	
Actors	Admin	
Description	The Admin edits or deletes existing librarian accounts to maintain system access control.	
Goal	To successfully add a new librarian account, enabling book and user management capabilities.	
Precondition	The Admin must be logged into the system and have access to the librarian management section. At least one librarian account must exist.	
Basic flow action	actor	System response
	1. The Admin navigates to the librarian management section. 2. The system displays a list of existing librarians. 3. The Admin selects a librarian to edit or delete.	Edit Path: - The system displays a form with the librarian's details. - The Admin updates fields (e.g., email, name) and clicks "Save." - The system validates the input and updates the database. - The system displays a confirmation message: "Librarian updated successfully." Delete Path: - The Admin confirms the deletion action. - The system removes the account from the database. - The system displays a confirmation message: "Librarian deleted successfully."
Postcondition	The selected librarian account is either updated with new details or deleted from the database.	
Alternative Actions	- If the updated input data is invalid (e.g., invalid email format or missing fields), the system displays an error message and prompts the Admin to correct the information. - If no librarians exist, the system displays an error message and prompts the Admin to return to the dashboard. - If the database fails to update or delete the account, the system displays an error message and prompts the Admin to try again.	

Table 10: Use Case Documentation for manage librarian

Use Case ID	UC#6	
Use case name	Register User	
Actors	Admin, Librarian	
Description	The Admin or Librarian creates a new user (student) account to allow access to the system’s book-related features.	
Goal	To successfully add a new user account, enabling the user to search, read, and download books.	
Precondition	The actor must be logged into the system and have access to the user registration section.	
Basic flow action	actor	System response
	1. The actor navigates to the user registration section on their dashboard. 2. The system displays a form requiring details such as name, email, username, and password. 3. The actor fills in the form and clicks "Submit."	4. The system validates the input for uniqueness and format. 5. If valid, the system creates the new user account in the database. 6. The system displays a confirmation message: "User registered successfully."
Postcondition	A new user account is created and stored in the database, ready for use.	
Alternative Actions	- If the input data is invalid (e.g., duplicate username or missing fields), the system displays an error message and prompts the actor to correct the information. - If the email is already registered, the system displays an error message and prompts the actor to use a different email. - If the database fails to save the account, the system displays an error message and prompts the actor to try again	

Table 11: Use Case Documentation for register user

Use Case ID	UC#7	
Use case name	Manage User Info	
Actors	Admin, Librarian	
Description	The Admin or Librarian edits or deletes existing user accounts to maintain user access control.	
Goal	To update or remove a user account to reflect current system access requirements.	
Precondition	The actor must be logged into the system and have access to the user management section. At least one user account must exist.	
Basic flow action	actor	System response
	1. The actor navigates to the user management section. 2. The system displays a list of registered users. 3. The actor selects a user to edit or delete.	Edit Path: - The system displays a form with the user's details. - The actor updates fields (e.g., email, name) and clicks "Save." - The system validates the input and updates the database. - The system displays a confirmation message: "User updated successfully." Delete Path: - The actor confirms the deletion action. - The system removes the account from the database. - The system displays a confirmation message: "User deleted successfully."
Postcondition	The selected user account is either updated with new details or deleted from the database.	
Alternative Actions	- If the updated input data is invalid (e.g., invalid email format or missing fields), the system displays an error message and prompts the actor to correct the information. - If no users exist, the system displays an error message and prompts the actor to return to the dashboard. - If the database fails to update or delete the account, the system displays an error message and prompts the actor to try again.	

Table 12: Use Case Documentation for manage user

Use Case ID	UC#8	
Use case name	Store Books Online	
Actors	Admin, Librarian	
Description	The Admin or Librarian adds a new book to the system’s inventory, making it available for users to access.	
Goal	To successfully add a new book to the inventory, enabling users to search, read, or download it.	
Precondition	The actor must be logged into the system and have access to the book management section.	
Basic flow action	actor	System response
	1. The actor navigates to the book management section. 2. The actor selects the "Add Book" option. 3. The system displays a form requiring book details (title, author, department, file upload). 4. The actor fills in the form, uploads the book file (e.g., PDF), and clicks "Submit.	5. The system validates the input (e.g., required fields, valid file format). 6. If valid, the system adds the book to the inventory and stores the file in the database. 7. The system triggers a notification to users and displays a confirmation message: "Book added successfully."
Postcondition	A new book is added to the inventory, stored in the database, and a notification is sent to users.	
Alternative Actions	- If the input data is invalid (e.g., missing title or unsupported file format), the system displays an error message and prompts the actor to correct the information. - If the book title already exists, the system displays an error message and prompts the actor to modify the title. - If the file upload fails, the system displays an error message and prompts the actor to retry the upload.	

Table 13: Use Case Documentation for store book online

Use Case ID	UC#9	
Use case name	Manage Books	
Actors	Admin, Librarian	
Description	The Admin or Librarian edits or deletes existing books in the inventory to maintain an accurate collection.	
Goal	To update or remove a book from the inventory to reflect current availability or details.	
Precondition	The actor must be logged into the system and have access to the book management section. At least one book must exist in the inventory.	
Basic flow action	actor	System response
	1. The actor navigates to the book management section. 2. The system displays a list of books in the inventory. 3. The actor selects a book to edit or delete.	Edit Path: - The system displays a form with the book's details. - The actor updates fields (e.g., title, author) and clicks "Save." - The system validates the input and updates the database. - The system displays a confirmation message: "Book updated successfully." Delete Path: - The actor confirms the deletion action. - The system removes the book from the database. - The system displays a confirmation message: "Book deleted successfully."
Postcondition	he selected book is either updated with new details or deleted from the database.	
Alternative Actions	- If the updated input data is invalid (e.g., missing required fields), the system displays an error message and prompts the actor to correct the information. - If no books exist, the system displays an error message and prompts the actor to return to the dashboard. - If the database fails to update or delete the book, the system displays an error message and prompts the actor to try again.	

Table 14: Use Case Documentation for manage book

Use Case ID	UC#10	
Use case name	Send Notification for New Book	
Actors	Admin, Librarian	
Description	The system automatically generates and sends a notification to all users when a new book is added to the inventory.	
Goal	To inform users about the availability of a new book, encouraging engagement with the system.	
Precondition	The actor must have successfully added a new book to the inventory via the Store Books Online use case.	
Basic flow action	actor	System response
	1. The actor adds a new book through the book management section. 2. The system generates a notification with details (e.g., book title, author). 3. The system stores the notification in the database with a status of "active."	4. The system sends the notification to all registered users. 5. Upon user login, the notification is displayed on their dashboard.
Postcondition	A notification is created, stored, and sent to all users, visible on their dashboard upon login.	
Alternative Actions	- If no users are registered, the system displays a message and prompts the actor to store the notification for future users. - If the notification content is invalid (e.g., missing book details), the system displays an error message and prompts the actor to review the book details. - If the notification fails to send, the system displays an error message and prompts the actor to retry.	

Table 15: use case documentation for Send Notification

Use Case ID	UC#11	
Use case name	Approve Comment	
Actors	Admin, Librarian	
Description	The Admin or Librarian reviews and approves user-submitted comments on books to ensure appropriate content is displayed.	
Goal	To approve a user comment, making it visible on the book's details page for other users to view.	
Precondition	The actor must be logged into the system and have access to the comment management section. At least one pending comment must exist.	
Basic flow action	actor	System response
	1. The actor navigates to the comment management section. 2. The system displays a list of pending comments. 3. The actor selects a comment to review. 4. The actor clicks "Approve.	5. The system updates the comment status to "approved" in the database. 6. The system displays a confirmation message: "Comment approved successfully."
Postcondition	The selected comment is approved and visible on the book's details page.	
Alternative Actions	-- If the actor chooses to reject the comment, the system displays a confirmation message and prompts the actor to remove the comment from the database. - If no comments are pending, the system displays an error message and prompts the actor to return to the dashboard. - If the database fails to update the comment status, the system displays an error message and prompts the actor to try again.	

Table 16: use case documentation for approve comment

Use Case ID	UC#12	
Use case name	Change Password	
Actors	Admin, Librarian, User (Student)	
Description	The actor changes their account password to maintain security or recover access.	
Goal	To successfully update the actor's password, ensuring continued secure access to the system.	
Precondition	The actor must be logged into the system and have access to the profile settings section.	
Basic flow action	actor	System response
	1. The actor navigates to the profile settings section. 2. The system displays a form for changing the password (current password, new password, confirm new password). 3. The actor fills in the form and clicks "Submit."	4. The system validates the input (e.g., current password matches, new password meets requirements). 5. If valid, the system updates the password in the database. 6. The system displays a confirmation message: "Password changed successfully."
Postcondition	The actor's password is updated in the database, and they can use the new password for future logins.	
Alternative Actions	- If the current password is incorrect, the system displays an error message and prompts the actor to re-enter the current password. - If the new password and confirmation do not match, the system displays an error message and prompts the actor to re-enter the new password. - If the database fails to update the password, the system displays an error message and prompts the actor to try again.	

Table 17: use case documentation for change password.

Use Case ID	UC#13	
Use case name	Read and Download Books	
Actors	Admin, Librarian, User (Student)	
Description	The actor accesses a book to read it online or download it as a digital file for offline use.	
Goal	To allow the actor to successfully read a book online or download it to their device.	
Precondition	The actor must be logged into the system and have access to the book catalog or search section. At least one book must be available in the inventory..	
Basic flow action	actor	System response
	1. The actor adds a new book 1. The actor navigates to the book catalog or search section. 2. The system displays a list of available books. 3. The actor selects a book from the list.	4. Read Online Path: - The actor clicks "Read Online." - The system loads the book content in the browser. - The actor reads the book. 4. Download Path: - The actor clicks "Download." - The system prompts the actor to save the file. - The actor saves the file to their device. 5. The system confirms the action (e.g., "Book loaded" or "Download successful").
Postcondition	The actor has either read the book online or downloaded it to their device for offline access.	
Alternative Actions	- If the book is unavailable, the system displays an error message and prompts the actor to select another book. - If the download fails (e.g., network issue), the system displays an error message and prompts the actor to retry the download. - If the file format is unsupported by the actor’s device, the system displays an error message and prompts the actor to use a compatible device.	

Table 18: use case documentation for read download book

Use Case ID	UC#14	
Use case name	Book Ratings	
Actors	User (Student)	
Description	The User submits a rating for a book to share feedback on its quality or relevance.	
Goal	To allow the User to successfully submit a rating for a book, pending approval for display.	
Precondition	The User must be logged into the system and have access to a book's details page. The book must be available in the inventory.	
Basic flow action	actor	System response
	1. The User navigates to a book's details page. 2. The system displays options to rate the book (e.g., 1-5 stars). 3. The User selects a rating and clicks "Submit."	4. The system validates the input (e.g., rating within range). 5. If valid, the system saves the rating in the database with a status of "pending." 6. The system displays a confirmation message: "Rating submitted successfully."
Postcondition	The User's rating is saved in the database, awaiting approval by an Admin or Librarian.	
Alternative Actions	- If the rating input is invalid (e.g., no rating selected), the system displays an error message and prompts the User to select a rating. - If the User has already rated the book, the system displays an error message and prompts the User to return to the book's details page. - If the database fails to save the rating, the system displays an error message and prompts the User to try again.	

Table 19: use case documentation for book rate

Use Case ID	UC#15	
Use case name	Send Comment	
Actors	User (Student)	
Description	The User submits a comment on a book to provide feedback or insights.	
Goal	To allow the User to successfully submit a comment for a book, pending approval for display.	
Precondition	The User must be logged into the system and have access to a book's details page. The book must be available in the inventory.	
Basic flow action	actor	System response
	1. The User navigates to a book's details page. 2. The system displays a comment submission form. 3. The User enters a comment and clicks "Submit."	4. The system validates the input (e.g., comment not empty, within character limit). 5. If valid, the system saves the comment in the database with a status of "pending." 6. The system displays a confirmation message: "Comment submitted successfully."
Postcondition	The User's comment is saved in the database, awaiting approval by an Admin or Librarian.	
Alternative Actions	- If the comment input is invalid (e.g., empty or exceeds character limit), the system displays an error message and prompts the User to correct the information. - If the User has already commented (if restricted), the system displays an error message and prompts the User to return to the book's details page. - If the database fails to save the comment, the system displays an error message and prompts the User to try again.	

Table 20: use case documentation for send comment

Use Case ID	UC#16	
Use case name	Logout	
Actors	Admin, Librarian	
Description	The actor ends their session to securely exit the system.	
Goal	To terminate the actor's session, ensuring no unauthorized access to their account.	
Precondition	The actor must be logged into the system with an active session.	
Basic flow action	actor	System response
	1. The actor navigates to their dashboard. 2. The actor locates and clicks the "Logout" option.	3. The system terminates the actor's session and clears session data. 4. The system redirects the actor to the login page.
Postcondition	The actor's session is ended, and they are redirected to the login page, requiring re-authentication for further access.	
Alternative Actions	- If the session has already expired, the system displays an error message and prompts the actor to log in again. - If the logout request fails due to a network issue, the system displays an error message and prompts the actor to try again.	

Table 21: use case documentation for logout

4.2. Object Model

The second step in system analysis

Object Model encompasses the principles of abstraction, encapsulation, modularity, hierarchy, typing, concurrency and persistence. Object Model basically emphasizes on the objectand class. And it represented in UML with class diagram.

4.2.1. Class Diagram

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints.

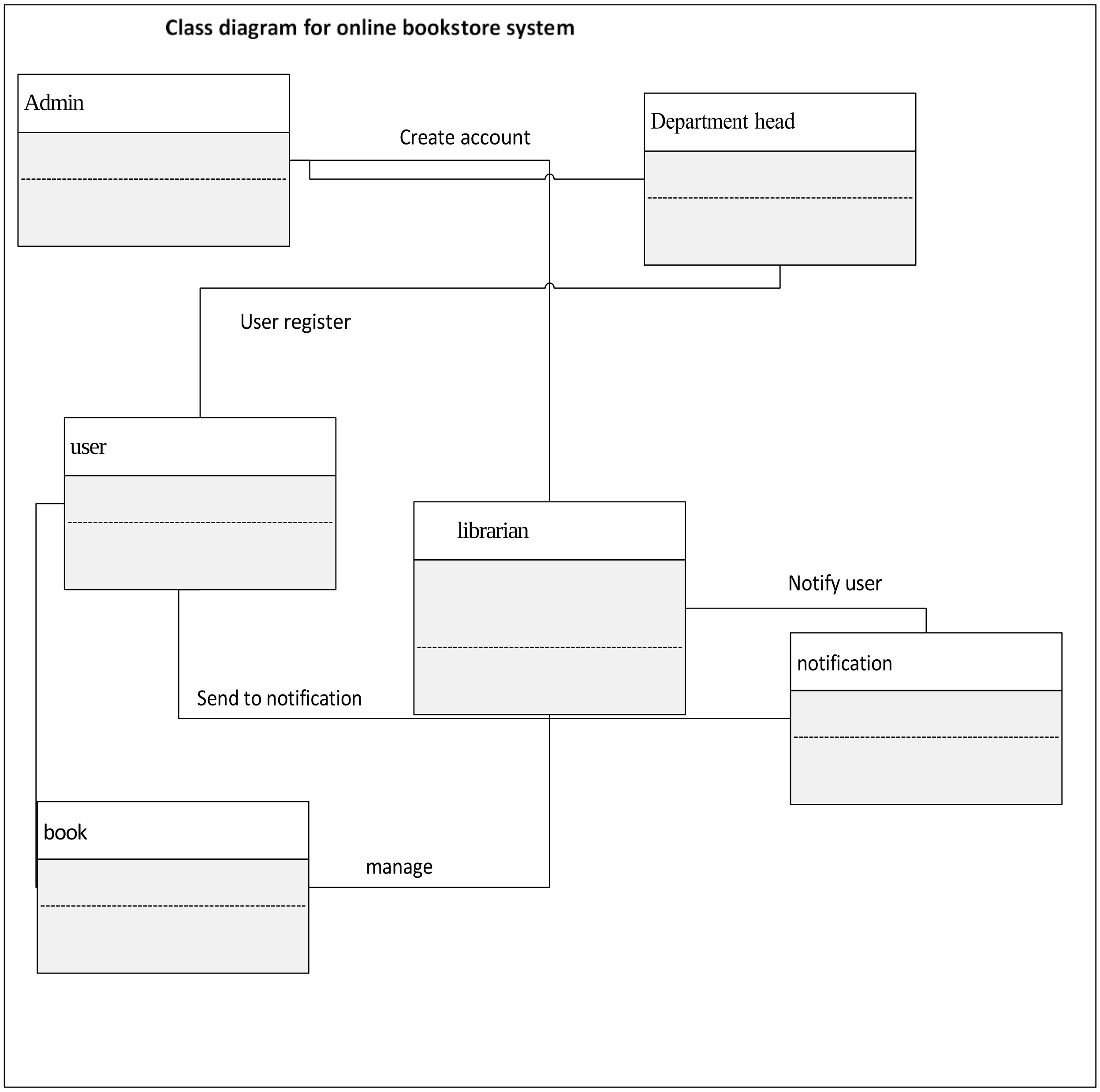


Figure 7: class diagram

4.2.2. Data Dictionary

Data Element	Definition	Type	Constraints	Field size
username	Unique name for logging in as an administrator.	String	Required, unique	20
password	Administrator's password	String	Required, minimum 6 characters	20
name	Full name of the administrator.	String	Required	30

Table 22: Data Dictionary for Administrator

Data Element	Definition	Type	Constraints	Field size
username	Unique name for logging in as a Department Head.	String	Required, unique	20
password	Department head password for authentication	String	Required, minimum 6 characters	20
name	Full name of the Department Head.	String	Required	20

Table 23: Data Dictionary for department head

Data Element	Definition	Type	Constraints	Field size
username	Unique name for logging in as a user.	String	Required, unique	20
password	user password for authentication	String	Required, minimum 6 characters	20
name	Full name of the user.	String	Required	20

Table 24: Data Dictionary for user

Data Element	Definition	Type	Constraints	Field size
username	Unique name for logging in as a Librarian.	String	Required, unique	20
password	Librarian's password for authentication.	String	Required, minimum 6 characters	20
name	Full name of the Librarian.	String	Required	30

Table 25: Data Dictionary for Librarian

Data Element	Definition	Type	Constraints	Field size
title	Title of the book.	String	Required	
author	Author of the book.	String	Required	50
department	Department or category of the book.	String	Optional	30
availability	Availability status of the book.	Boolean	Default: true	20

Table 26: Data Dictionary for Book

Data Element	Definition	Type	Constraints	Field size
Notification-id	Unique identifier for notification	integer	Required, unique	20
message	Content of notification	String	Required	200
Date creates	Date and time notification was created	Date	Required	30
status	Status of notification	string	required	15

Table 27: Data Dictionary for notification

4.3. Dynamic Model

The dynamic model is used to express and model the behavior of the system over time. It includes support for activity diagrams, state diagrams, and sequence diagrams.

4.3.1. Sequence Diagram

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

Here is sequence diagram for the proposed system

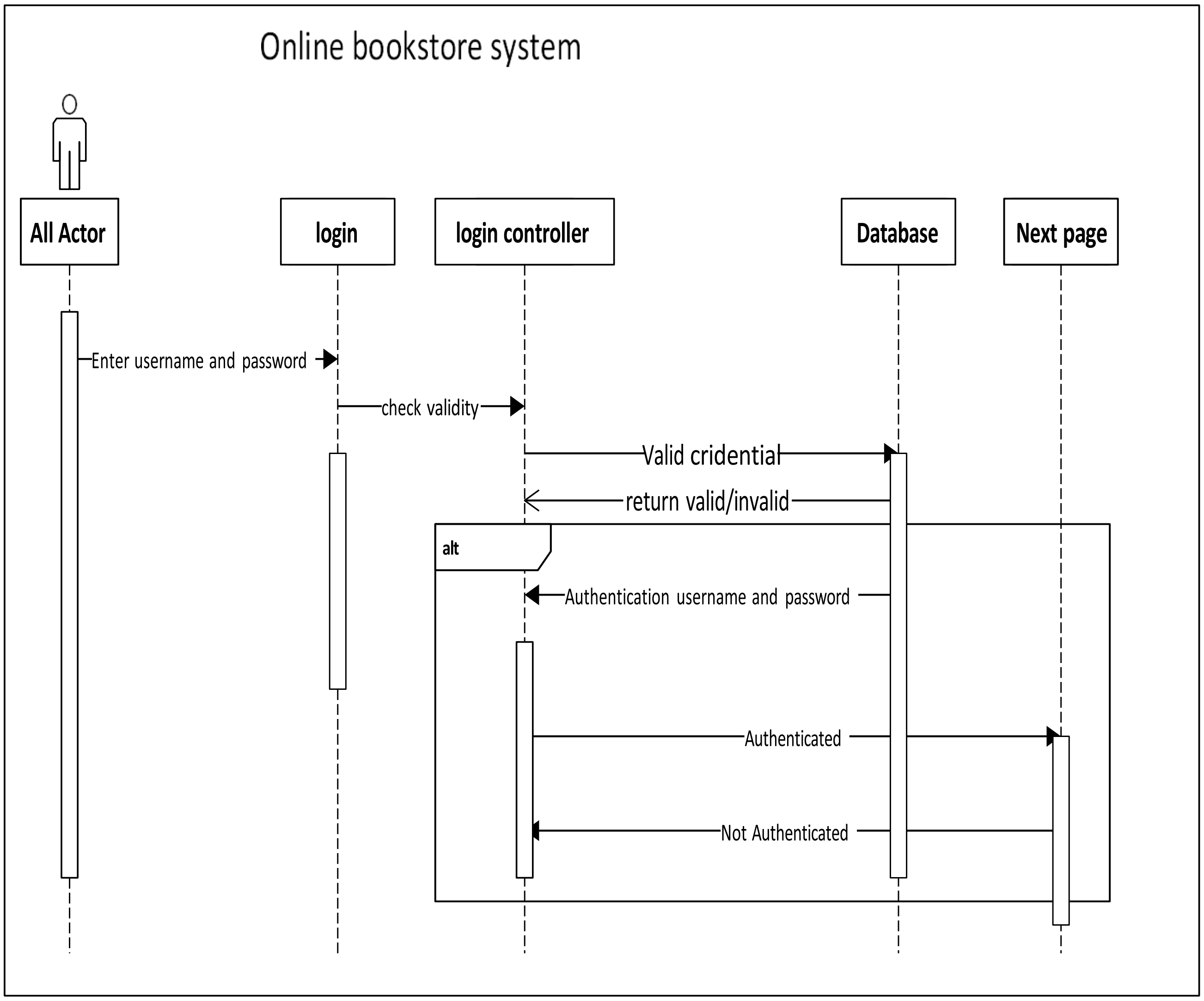


Figure 8: sequence diagram for login

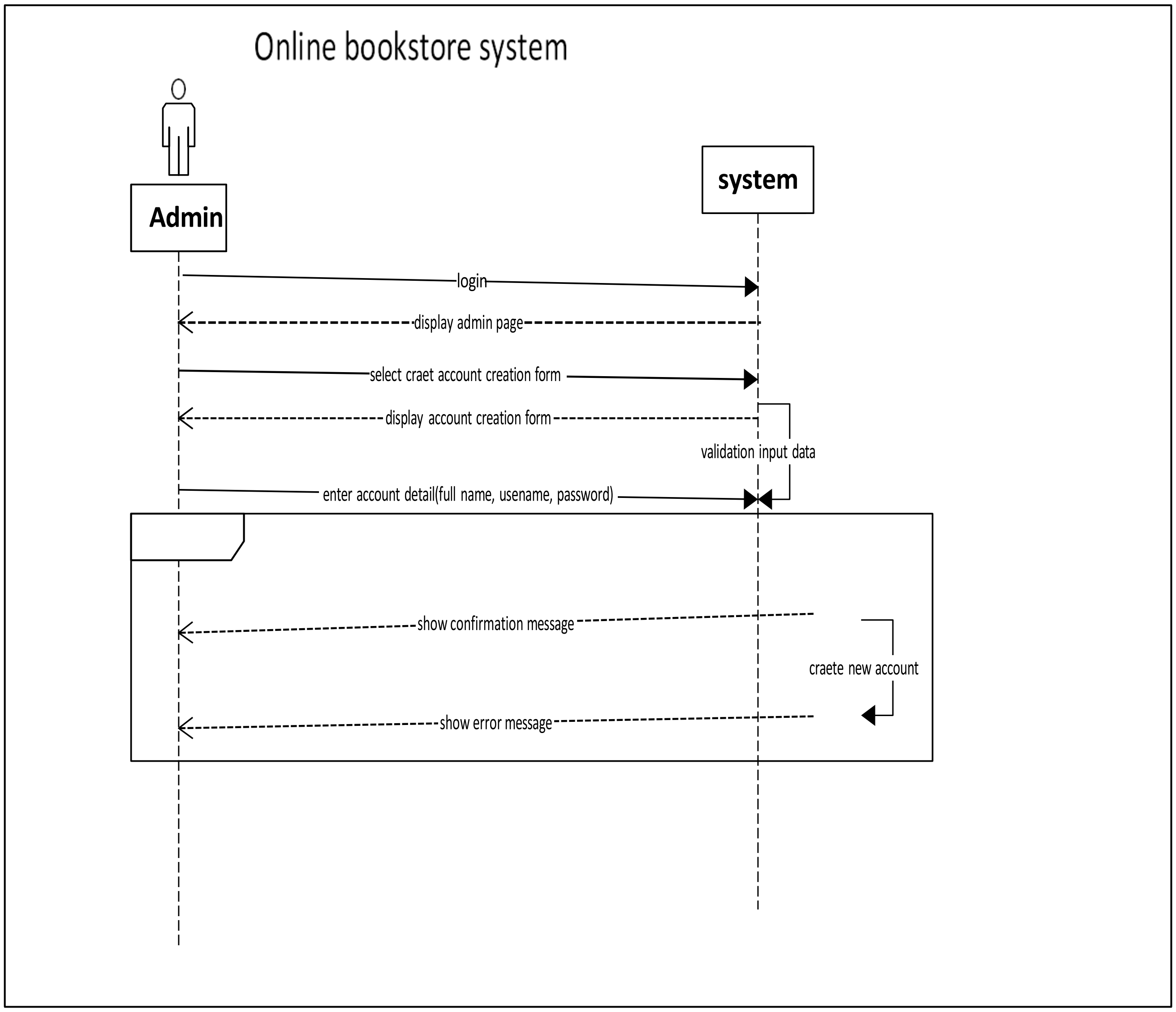

Online bookstore system
Admin
system
login
display admin page
select craet account creation form
display account creation form
validation input data
enter account detail(full name, usename, password)
show confirmation message
craete new account
show error message

Figure 9: sequence diagram for create account

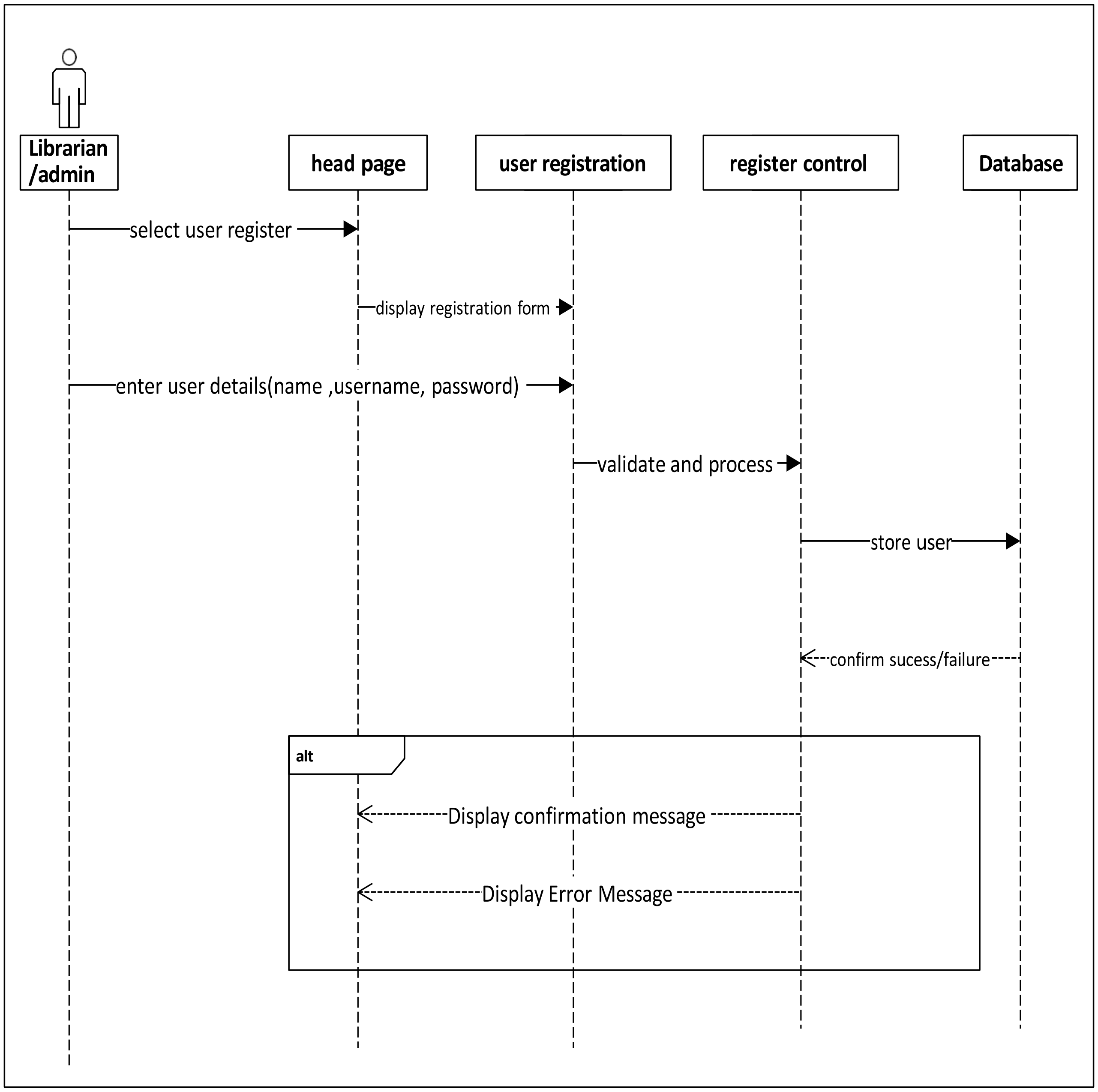


Figure 10: sequence diagram for user register

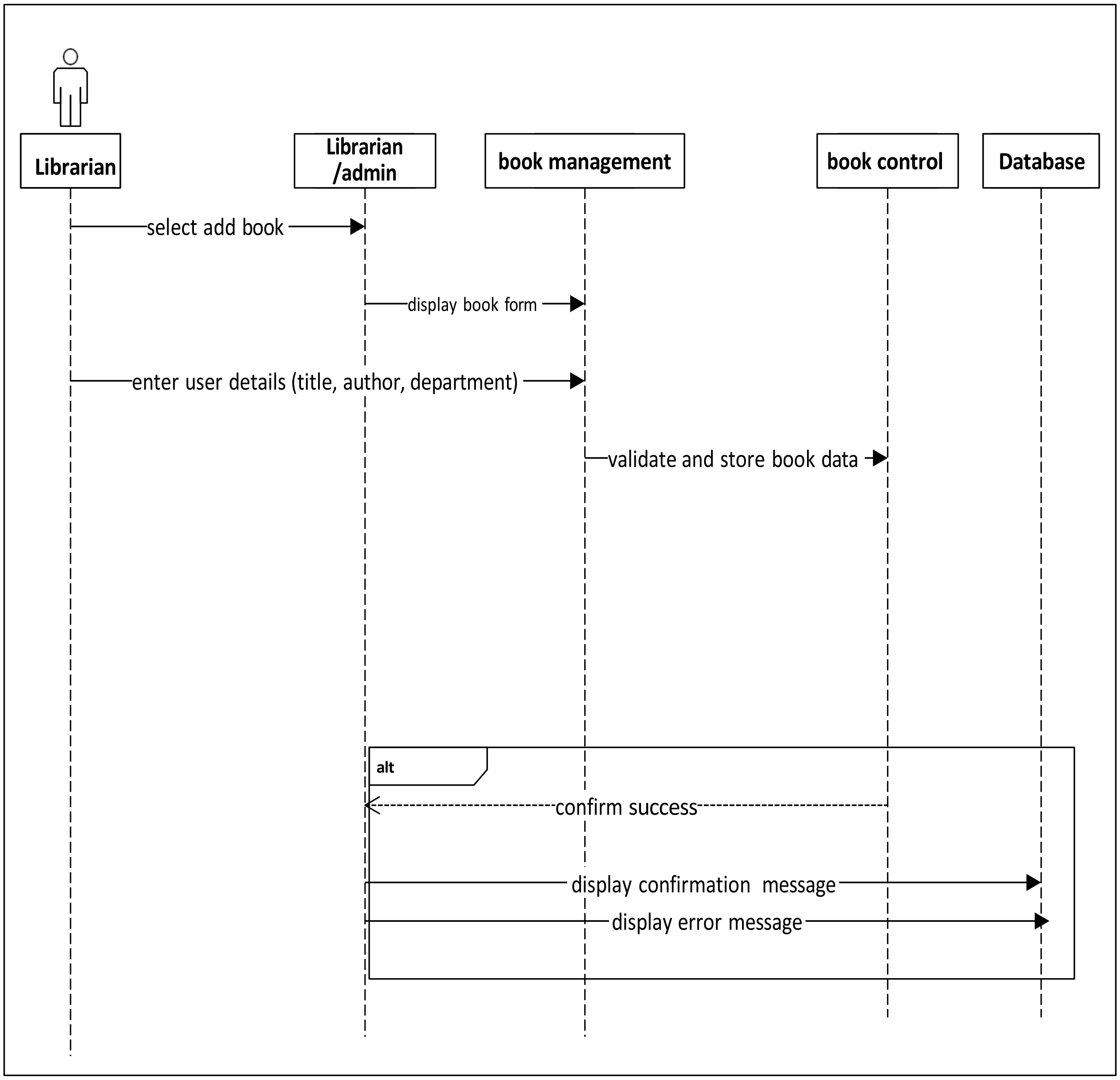


Figure 11: sequence diagram for store book online

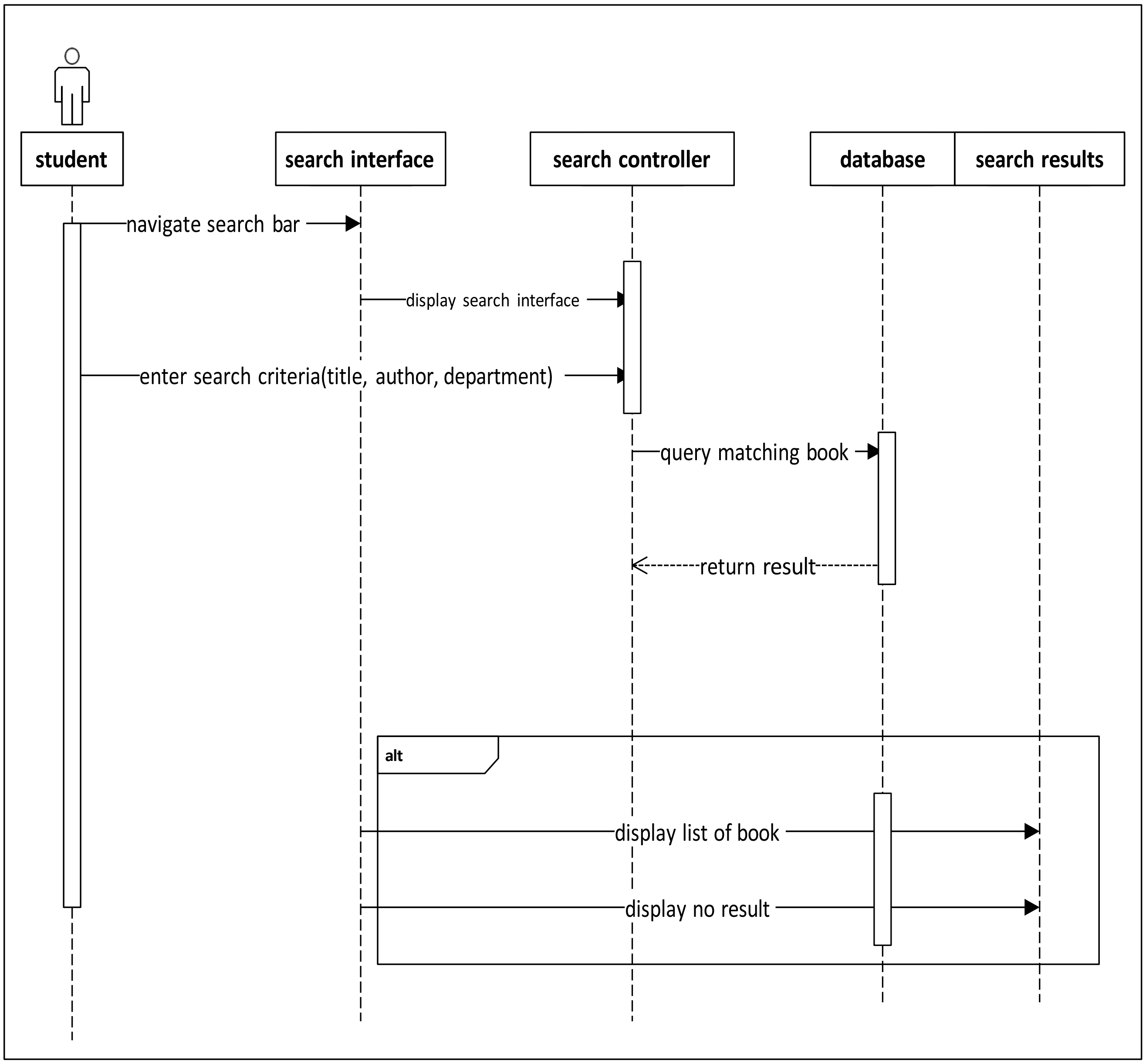


Figure 12: sequence diagram for search book

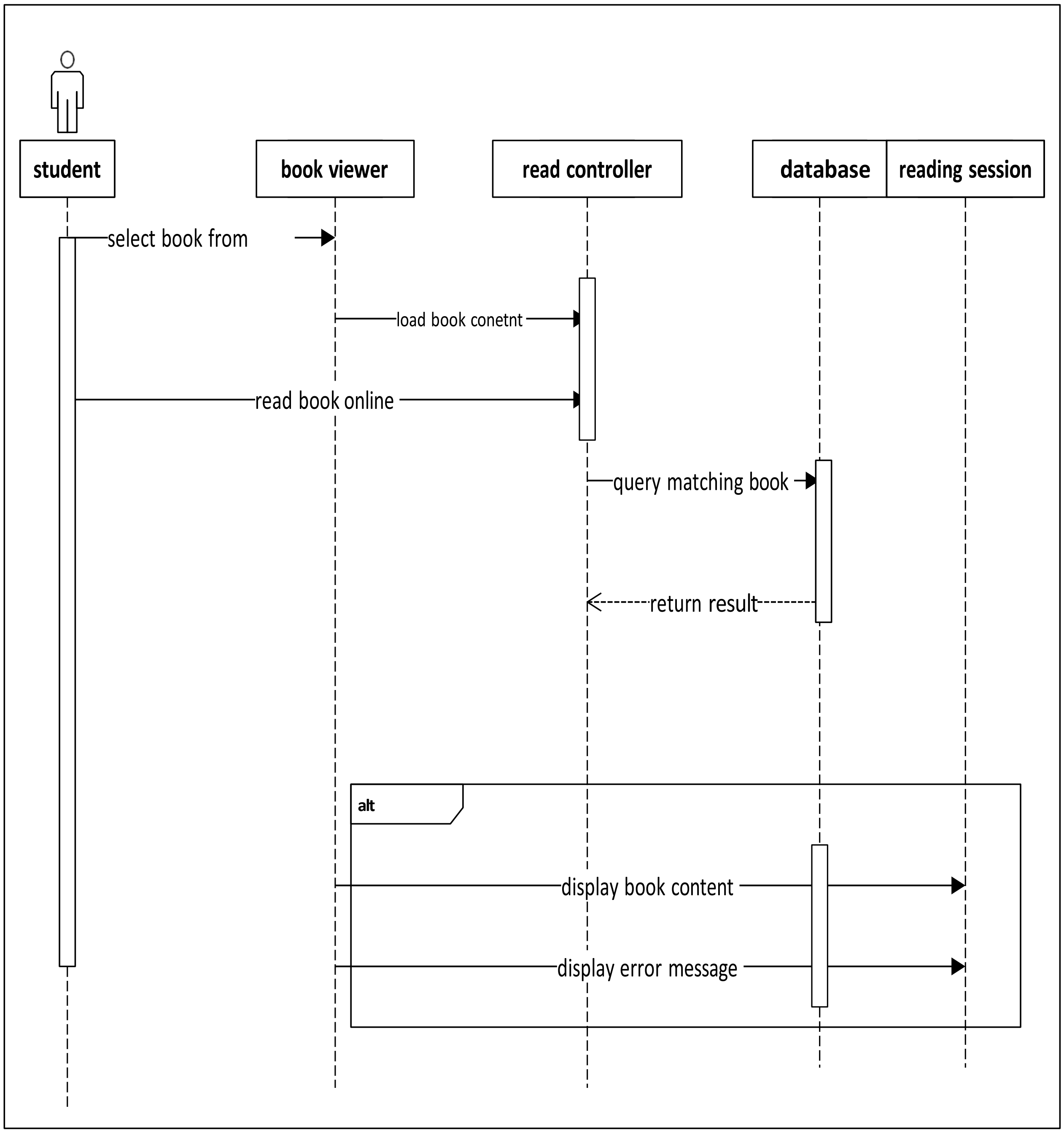


Figure 13: sequence diagram for read book online

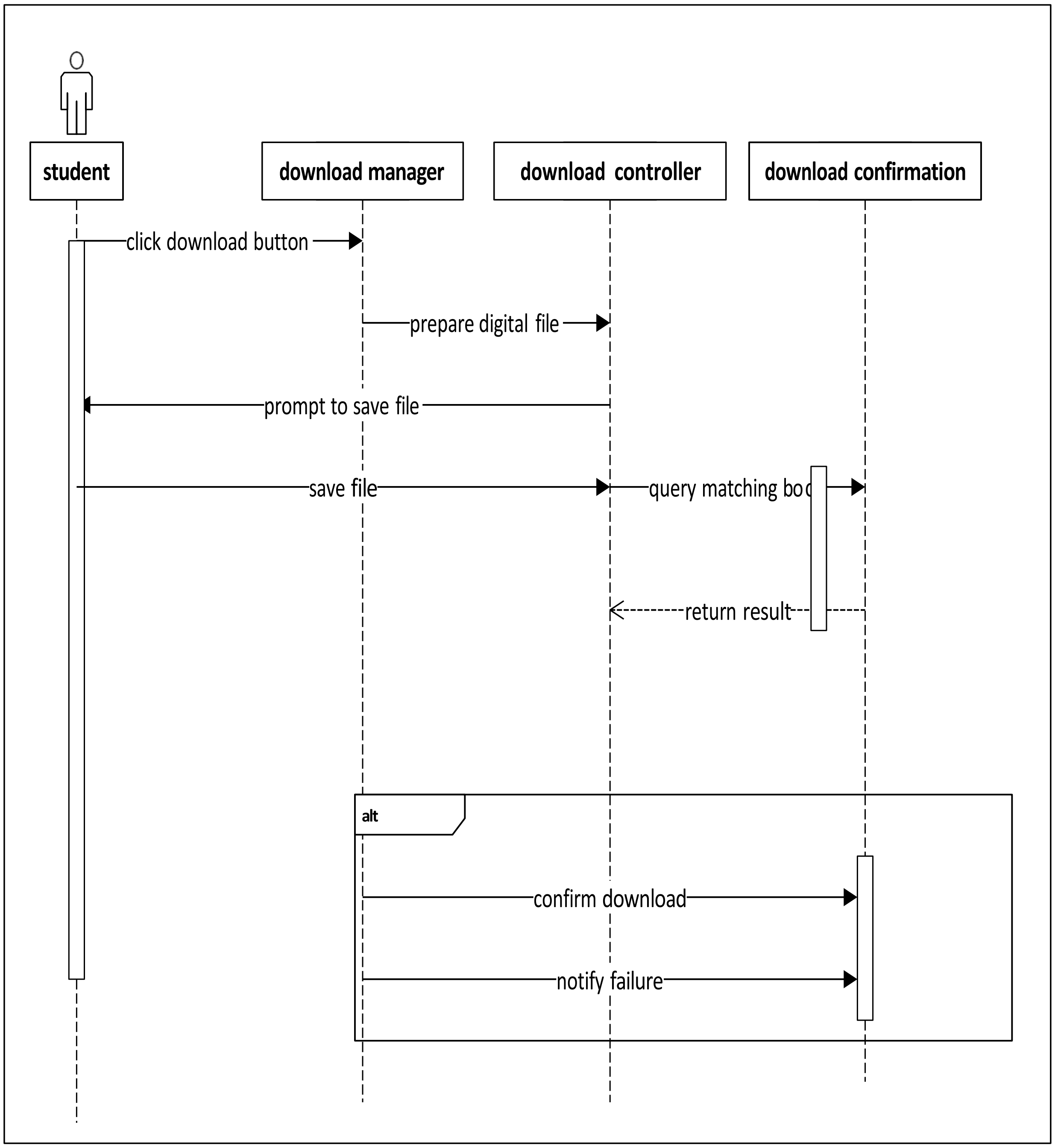


Figure 14: sequence diagram for download book

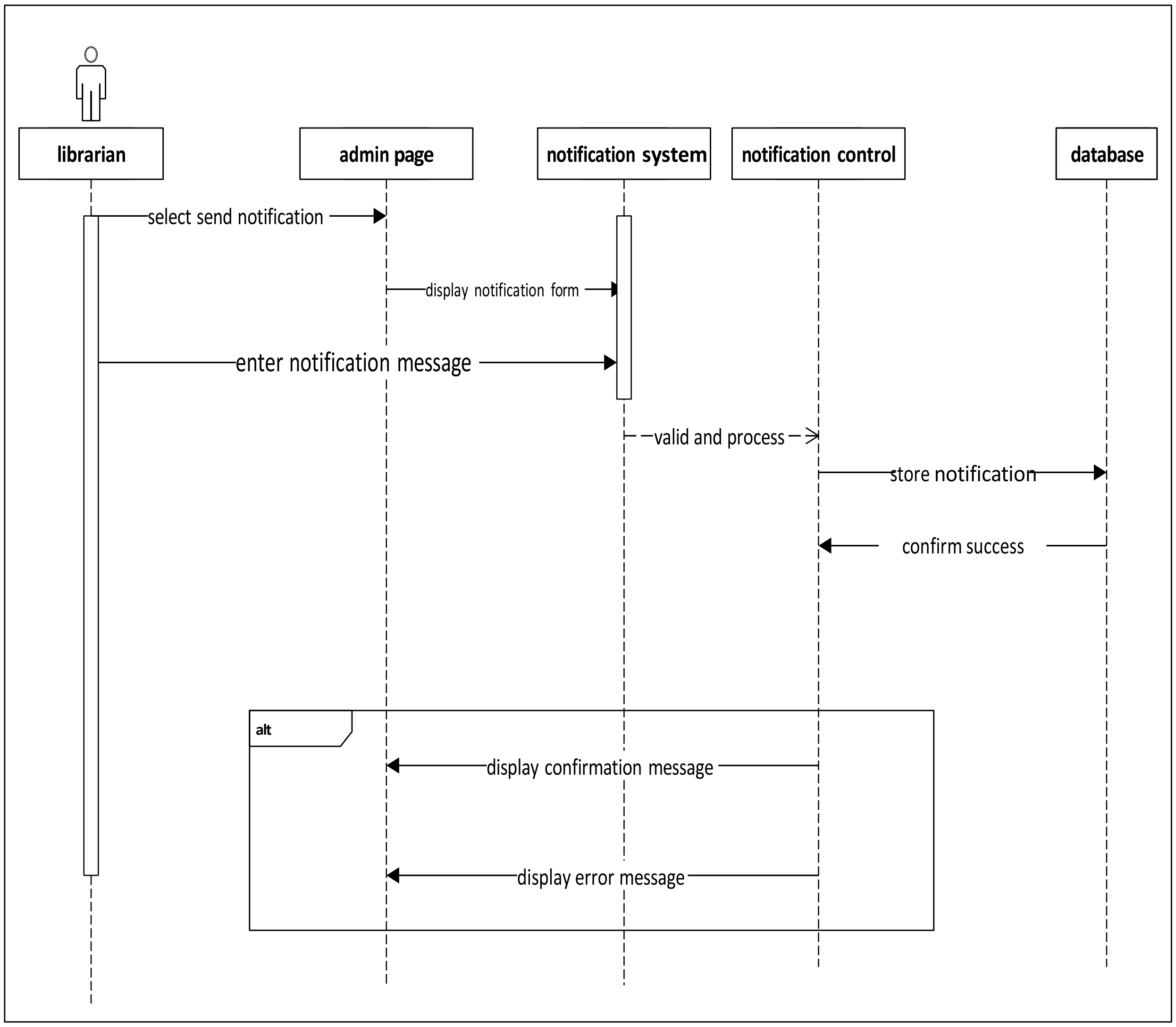


Figure 15: sequence diagram for notification

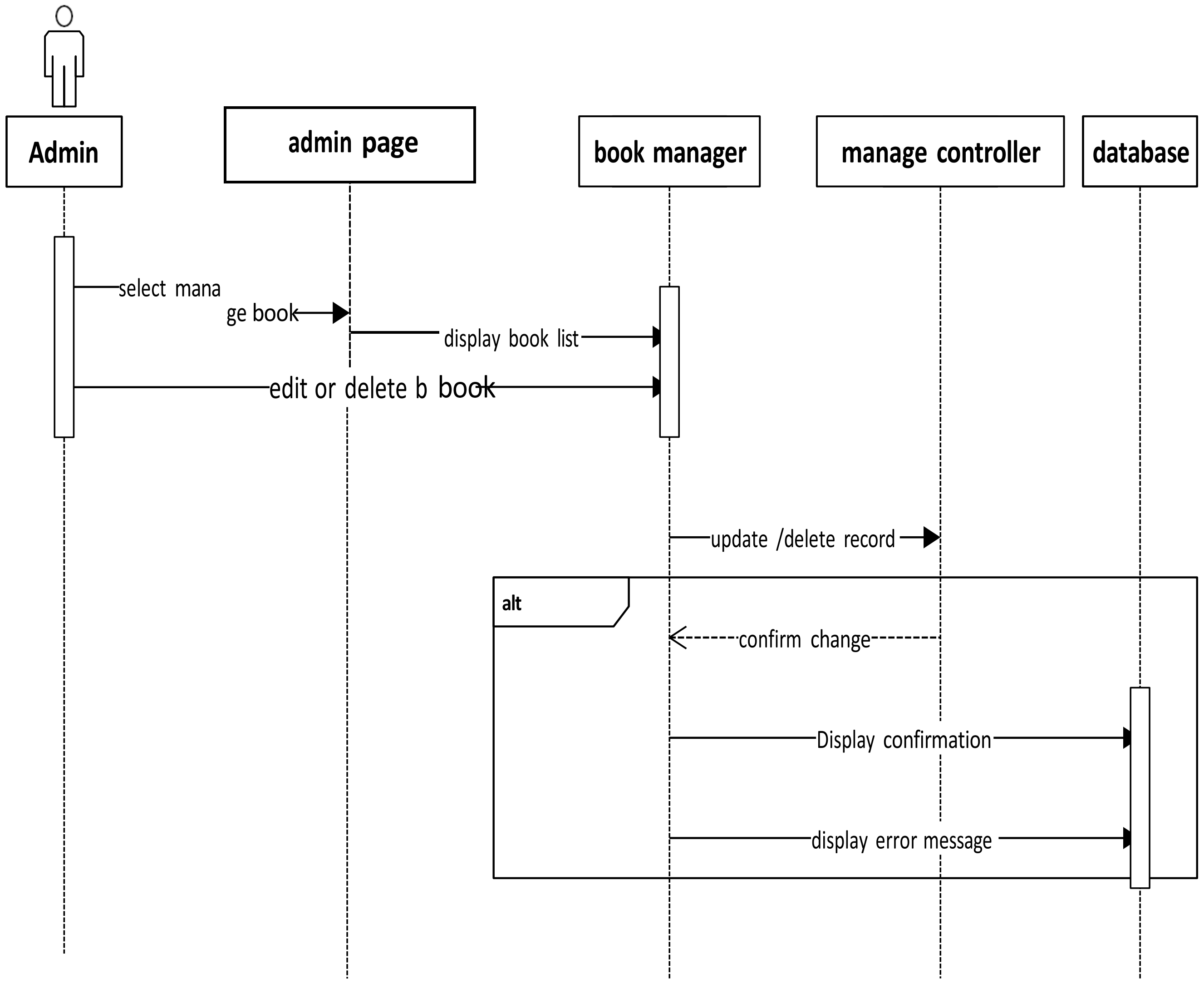


Figure 16: sequence diagram for manage book

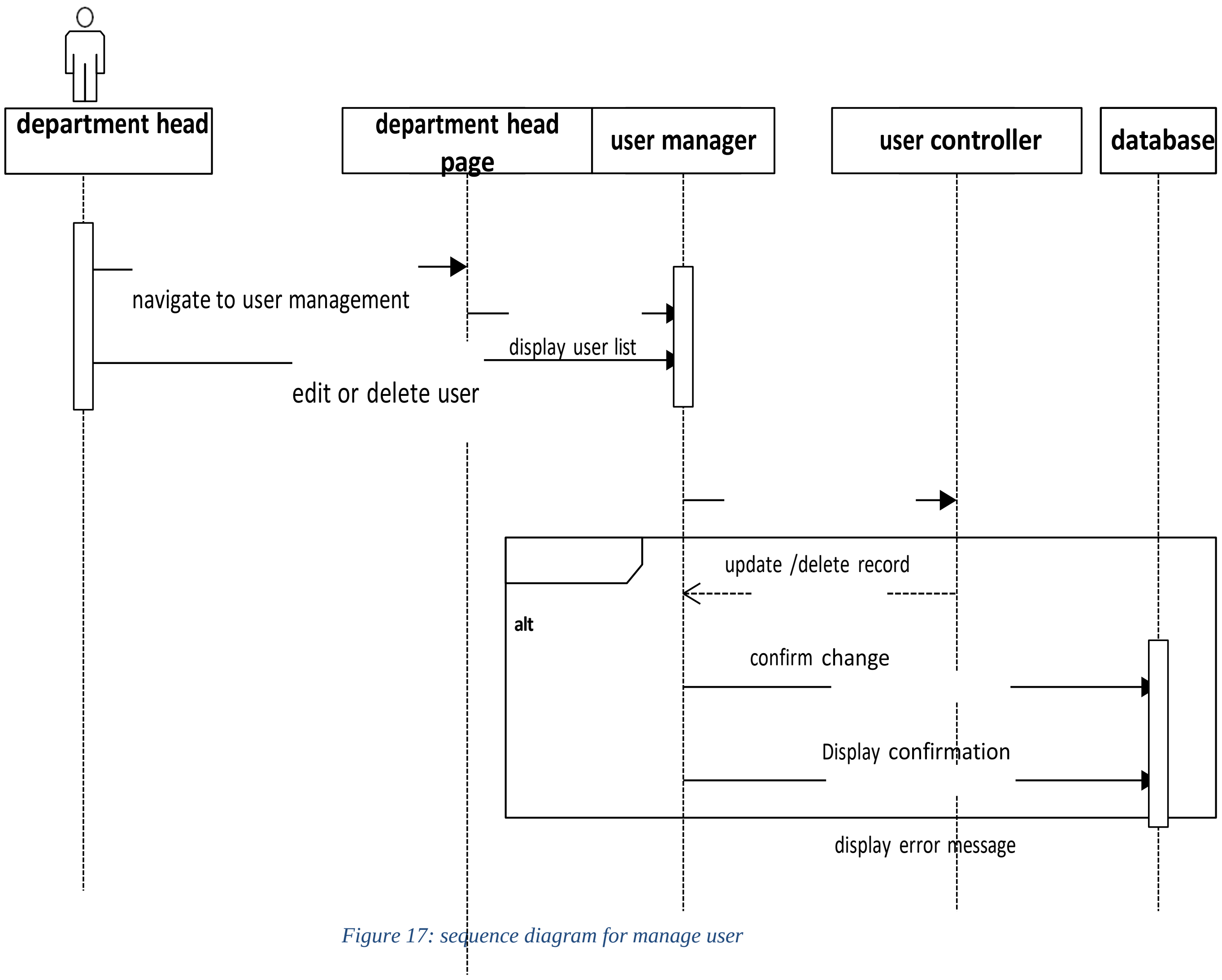


Figure 17: sequence diagram for manage user

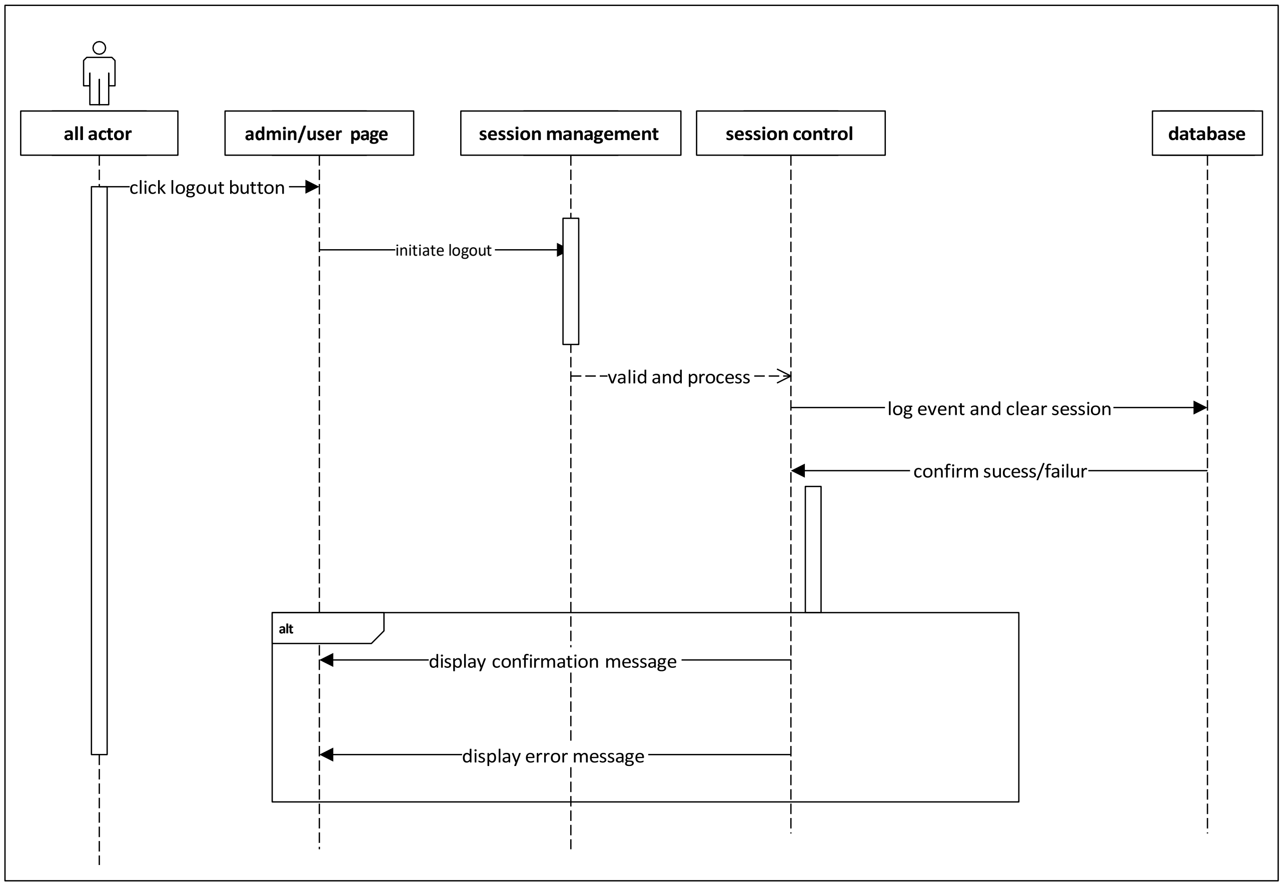


Figure 18: sequence diagram for logout

4.3.2. Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. Therefore, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

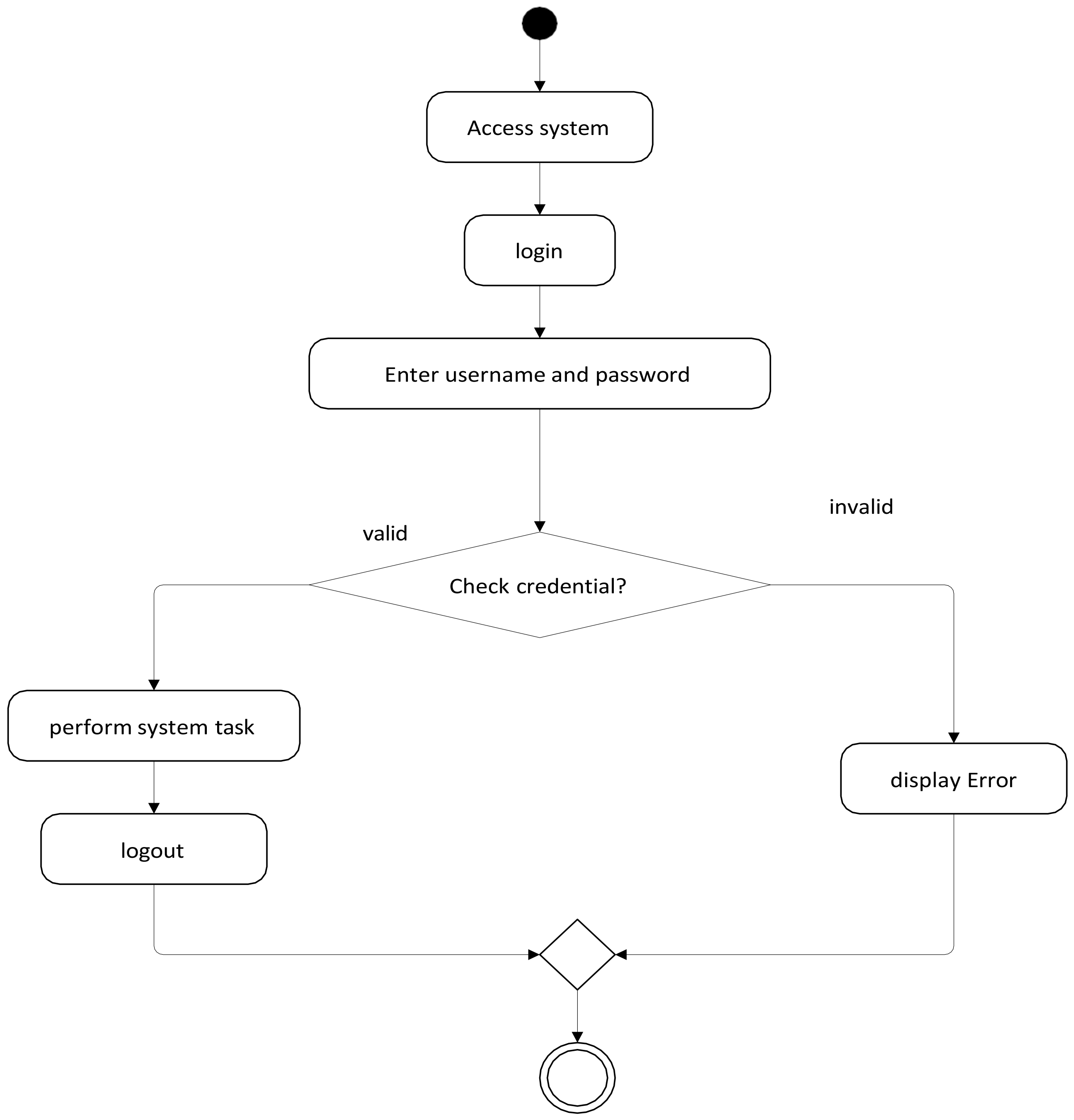


Figure 19: activity diagram

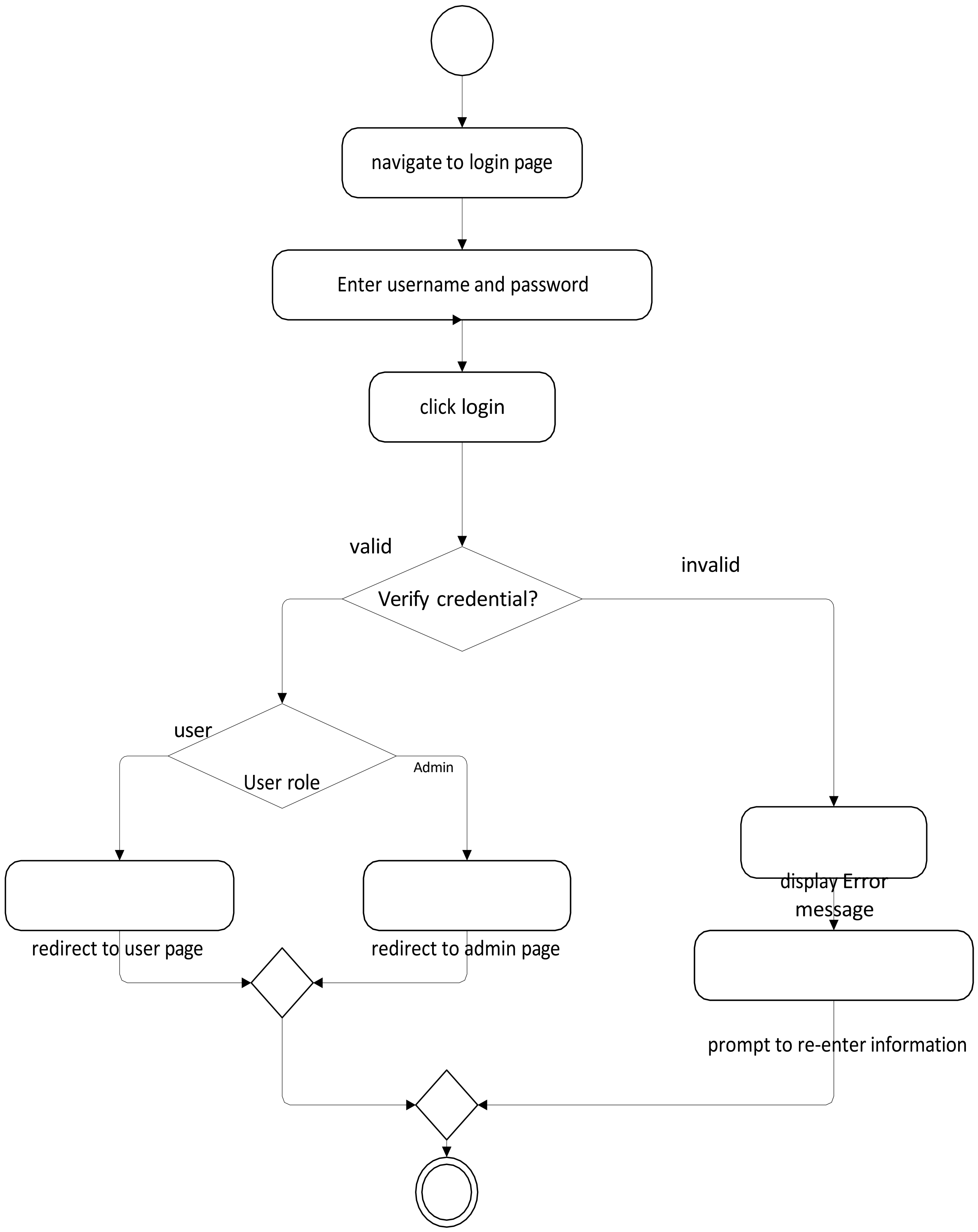


Figure 20: activity diagram for login

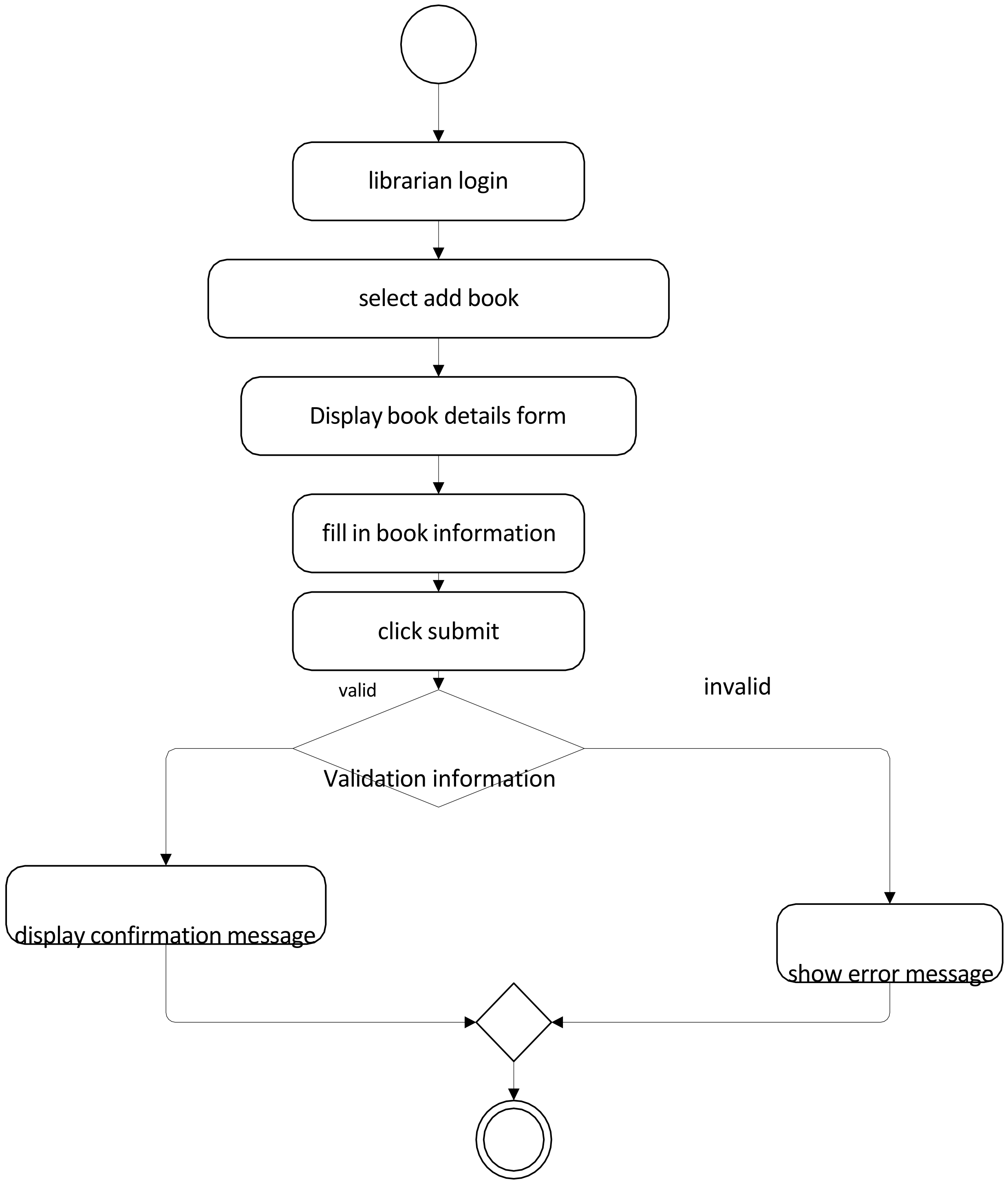


Figure 21: activity diagram for store book

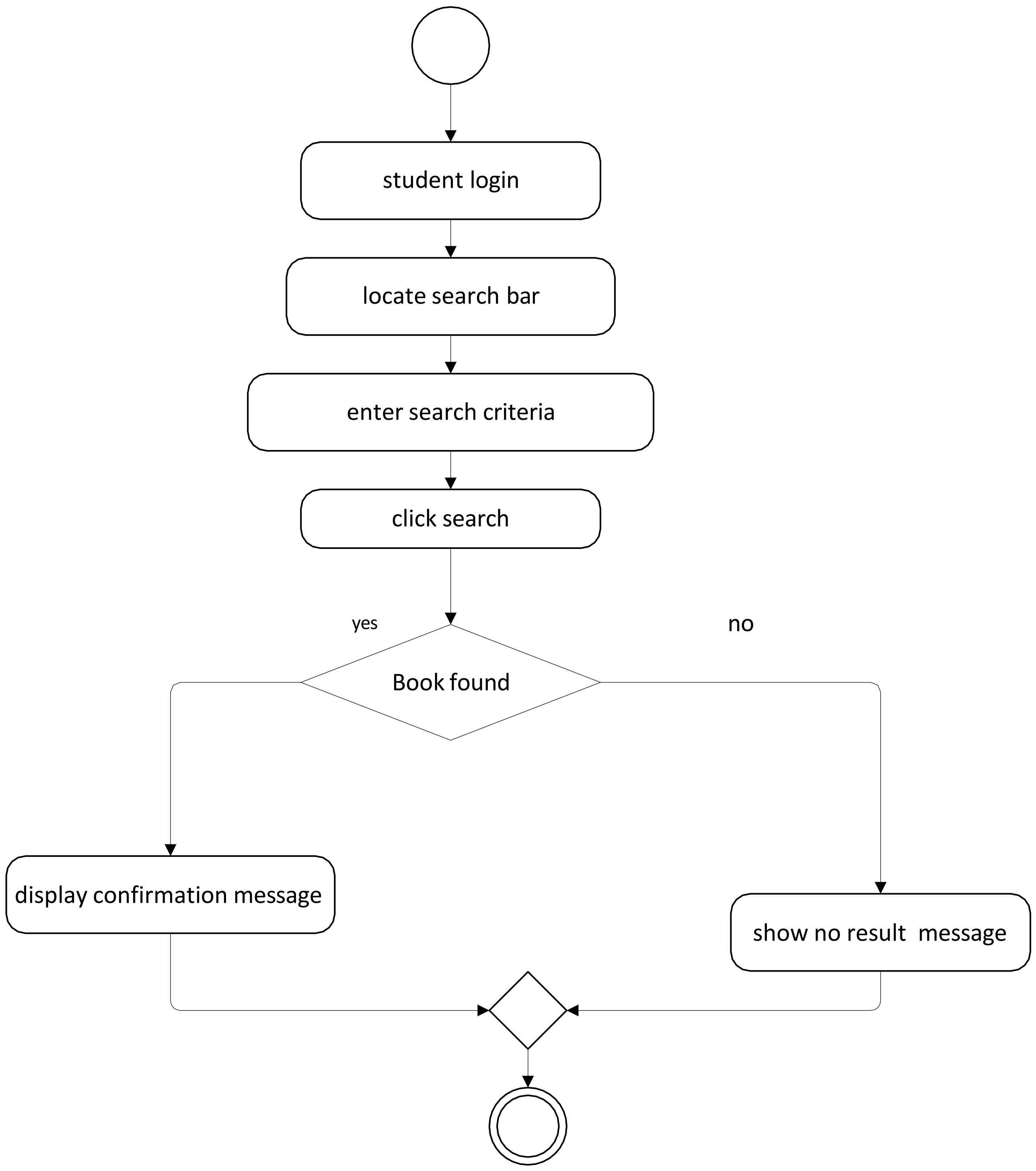


Figure 22: activity diagram for search book

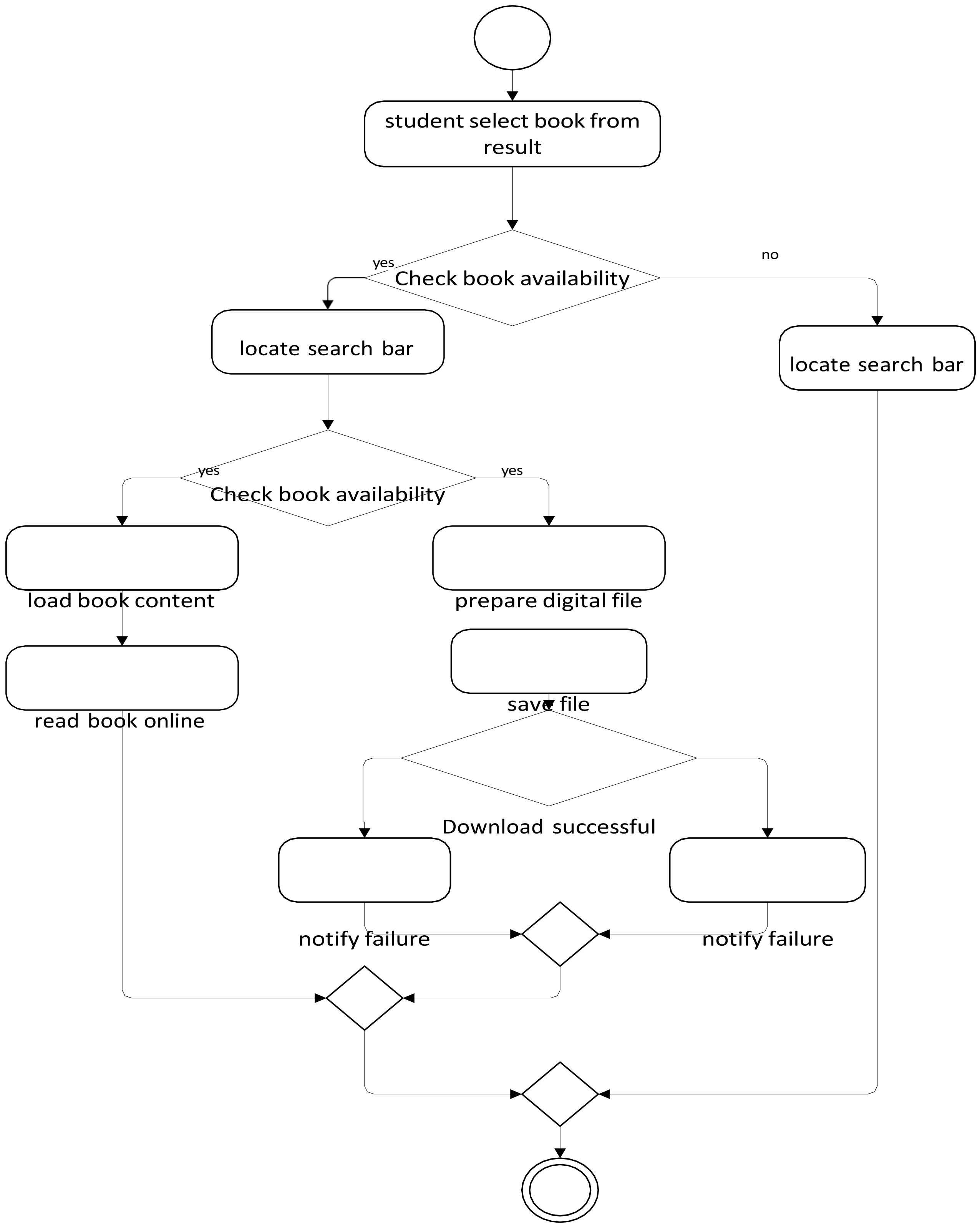


Figure 23: activity diagram for read and download book

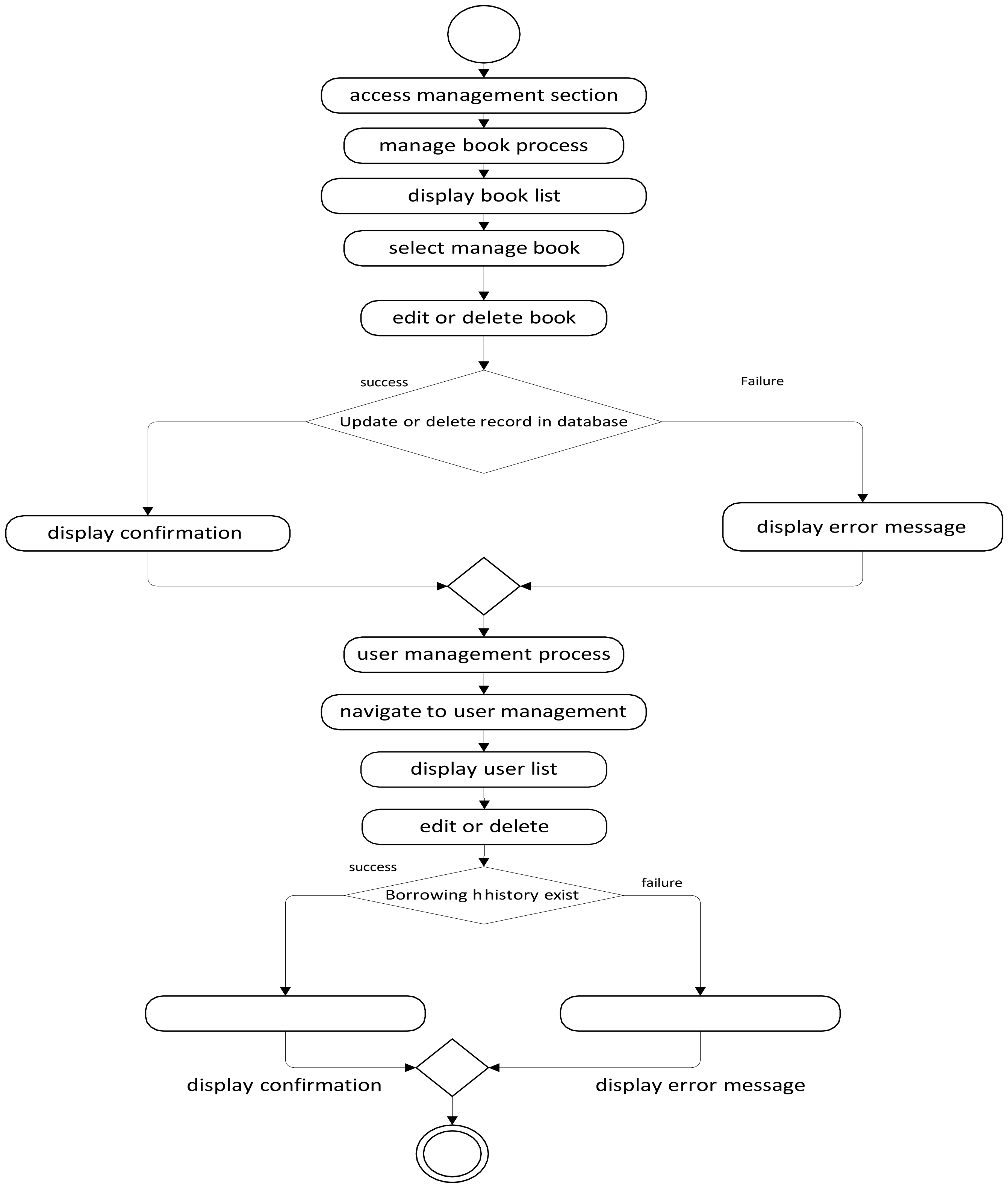


Figure 24: activity diagram for book and user management

4.3.3. State Chart Diagram

A state chart diagram is a view of a state machine that models the changing behavior of a state. State chart diagrams show the various states that an object goes through as well as the events that cause a transition from one state to another

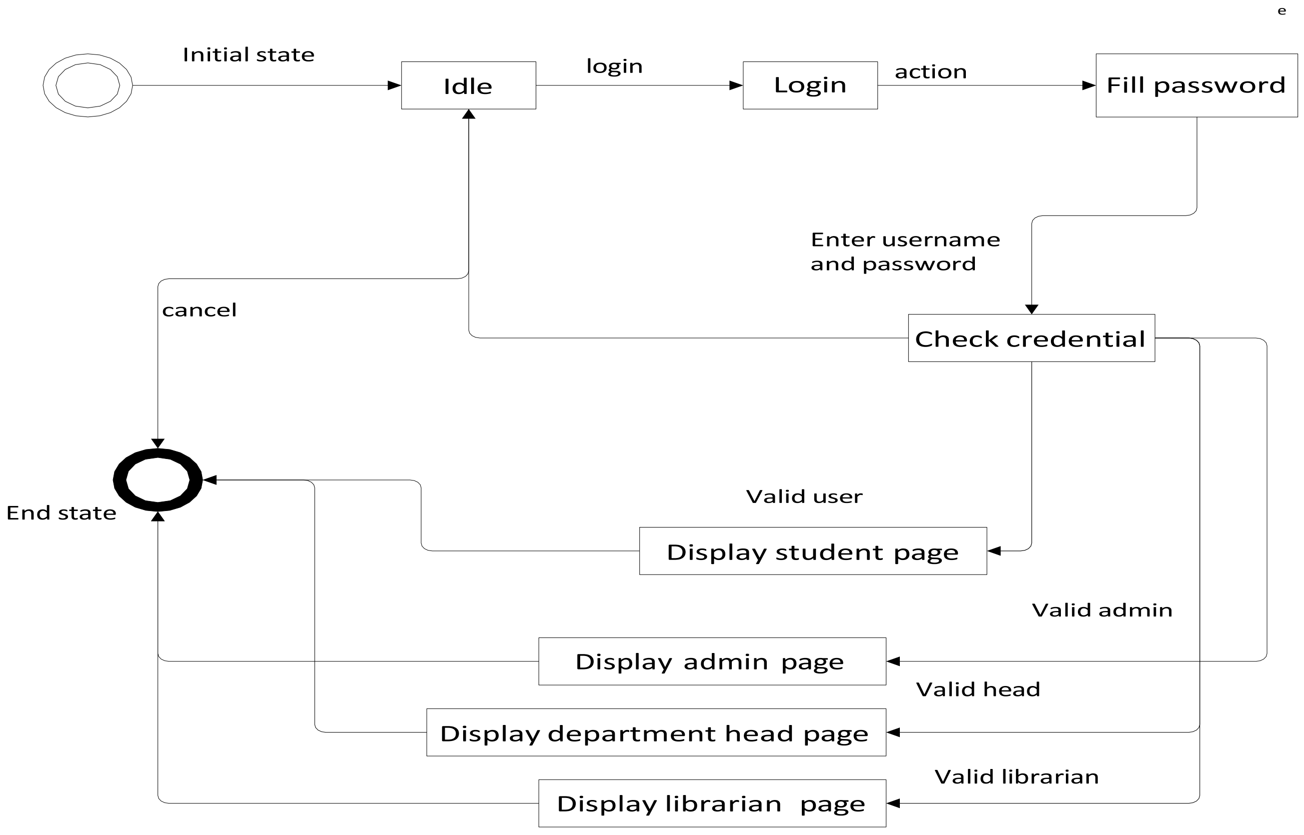


Figure 25:state chart diagram for login

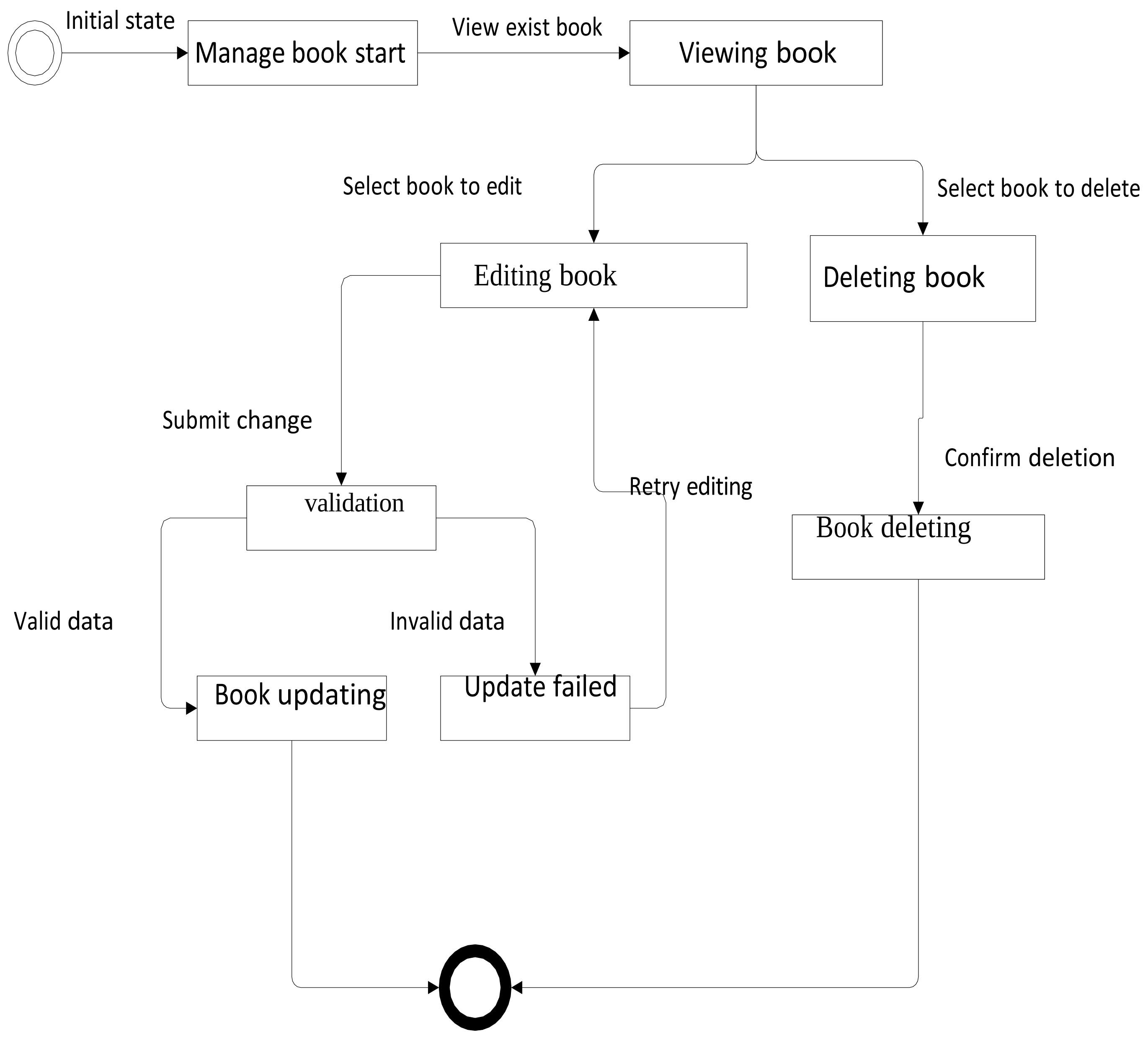


Figure 26: state chart diagram for manage book

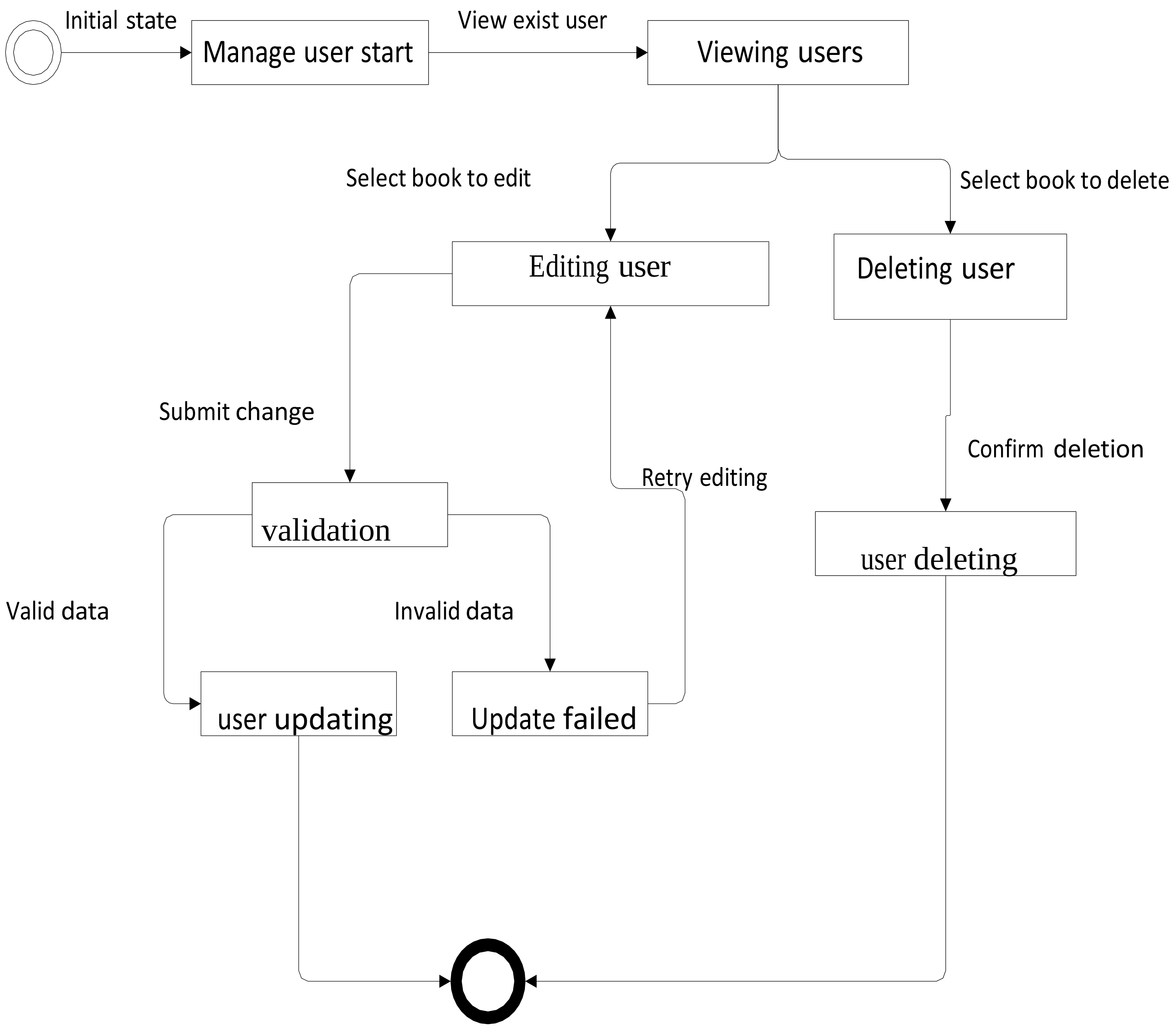


Figure 27:state chart diagram for user management

CHAPTER FIVE

5. SYSTEM DESIGN

5.1. Design Goals

The design goals are established to ensure that the system meets the needs of its users while providing an efficient and user-friendly experience.

1. Performance: The system should provide quick response times for all user interactions, including searching for books and downloading. It aims to minimize loading times and ensure smooth navigation, even during peak usage times.
2. Usability: A user-friendly interface is crucial. The design will focus on intuitive navigation, clear labeling of features, and accessible layout to accommodate users of varying technical skills. This will enhance the overall user experience and encourage engagement with the system.
3. Security: The system will implement robust security measures to protect user data and ensure that only authorized individuals have access to sensitive information. This includes secure authentication methods, data encryption, and role-based access controls for different user types (students, faculty, administrators).
4. Extensibility: The architecture will be designed with future enhancements in mind, allowing for the easy addition of new features and functionalities without disrupting existing operations. This will ensure the system remains relevant and adaptable to evolving user needs.
5. Maintainability: The code base will be structured to facilitate easy updates and maintenance. This includes using modular programming practices, clear documentation, and standardized naming conventions to ensure that future developers can efficiently manage and modify the system.
6. Accessibility: The design will ensure that the system is accessible to all users. This may involve compliance with web accessibility standards and providing alternative means of accessing content.

5.2. Proposed System Architecture

The proposed architecture for the Web-Based Online Book Store System is designed to ensure scalability, maintainability, and user accessibility. The system follows a three-tier architecture model, which separates the application into three distinct layers: the client layer, the application server layer, and the database layer. This separation enhances the management of the system and optimizes performance.

1. Client Layer:

- This layer consists of user interfaces accessible through web browsers on various devices, including desktops, laptops, tablets, and smartphones.

- Users interact with the system through a responsive web interface designed using HTML, CSS, and JavaScript, which allows for seamless navigation and access to features like searching for books, reading and downloading.

2. Application Server Layer:

- The application server handles the business logic of the system. It processes user requests, executes the necessary operations, and generates responses.
- This layer is implemented using PHP, which interacts with the database to retrieve or update information based on user actions.
- The server also manages user sessions, ensuring secure access and maintaining state throughout the user's interaction with the system.

3. Database Layer:

- The database layer is responsible for data storage and management. It utilizes MySQL as the database management system to store user profiles, book records and other necessary data.
- This layer ensures that data is organized efficiently, allowing for quick retrieval and updates. It incorporates security measures to protect sensitive information and supports real -time inventory tracking to provide users with accurate availability status.

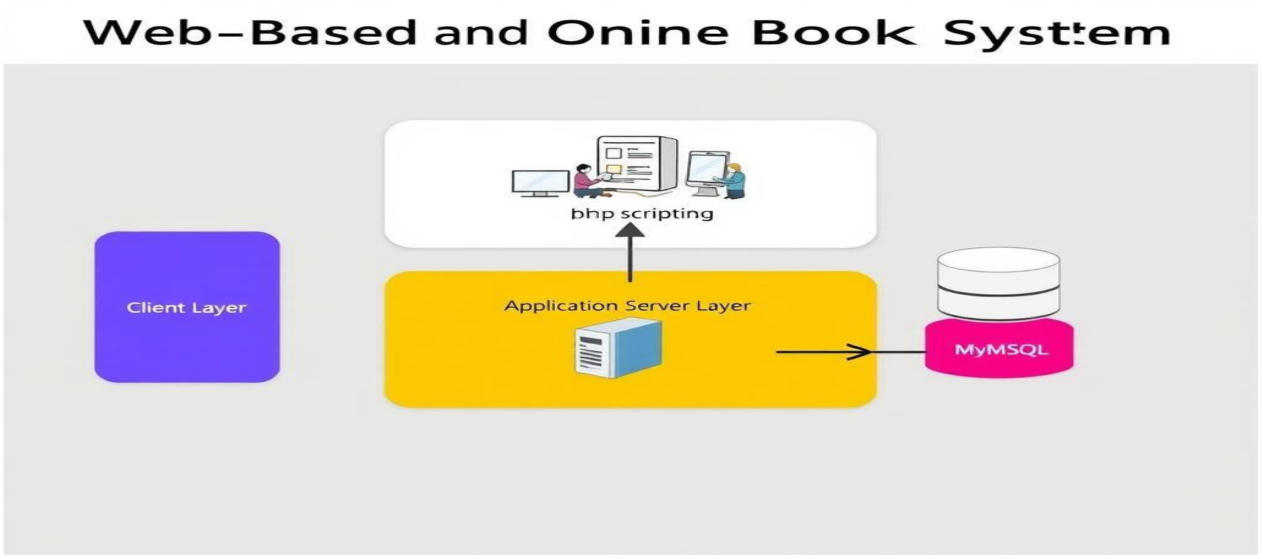


Figure 28: propose system architecture

5.2.1. Subsystem Decomposition and Description

Subsystem decomposition will help reduce the complexity of the system. The subsystems can be considered as packages holding related classes/objects. Our system is decomposed into: -

1. User Management Subsystem:

Description: This subsystem is responsible for handling user registration, authentication, and profile management. It ensures that users can create accounts, log in securely, and manage their personal information.

2. Book Management Subsystem:

Description: This subsystem manages the life-cycle of books in the online store, including adding new books, updating existing records, and deleting books that are no longer available. It ensures that the book inventory is up-to-date and accurately reflects available resources.

3. Search and Retrieval Subsystem:

Description: This subsystem enables users to search for books based on various criteria such as title, author, and department. It retrieves relevant book information from the database and presents it to the user in an easily navigable format.

4. Read and Downloading Subsystem:

Description: This subsystem manages the user process for digital copies of books, allowing users to read, and download books as needed.

5. Notification Subsystem:

Description: This subsystem handles notifications related to user actions, such as new book additions, and system updates. It ensures that users stay informed about their activities within the system.

5.2.2. Hardware/Software Mapping

The Web-Based Online Book Store System for ARU University is deployed using a client-server architecture, where hardware components support the software artifacts to ensure efficient operation. This mapping outlines the hardware and software required to deliver the system’s functionalities, including user management, book browsing, downloading, and inventory tracking.

1. **Server:**

- ➢ **Processor:** Intel Xeon or AMD Ryzen (multi-core) to handle multiple user requests, such as searches, downloads, and reading processes.
- ➢ **RAM:** Minimum 8 GB (scalable based on user load) to support simultaneous access by ARU students and administrators.
- ➢ **Storage:** SSD with at least 1 TB capacity for storing the database and digital book files (e.g., PDFs) for unlimited downloads.

2. **Client Devices:**

- ➢ PCs, laptops, tablets, and smartphones with modern web browsers, requiring at least 4 GB RAM and 64 GB storage (for downloaded books), plus a stable internet connection (minimum 5 Mbps) to access the system from anywhere (e.g., classrooms, dorms, home).

5.2.2.1 Software mapping

1. **Operating System:**

- **Server:** Linux (e.g., Ubuntu Server) or Windows Server to host the web and database servers securely and reliably.
- **Client Devices:** Windows, macOS, Linux, Android, or iOS to support browser-based access for students and administrators.

2. **Web Server:** Apache (via XAMPP) to serve web pages, manage user requests (e.g., login, search, download), and deliver book content efficiently.
3. **Database Management System (DBMS):**

- ➢ MySQL (via XAMPP) to store and manage persistent data, including user accounts, book details (title, author, department, availability).

4. **Backend Programming Language and Framework:**

- PHP to implement the application logic, such as processing searches by title/author/department, handling unlimited book downloads and updating inventory in real-time.

5. **Frontend Framework:**

- HTML, CSS, and JavaScript to build a responsive, user-friendly interface for browsing books, reading online, and managing user/admin tasks, ensuring compatibility across devices.

5.2.3. Detailed Class Diagram

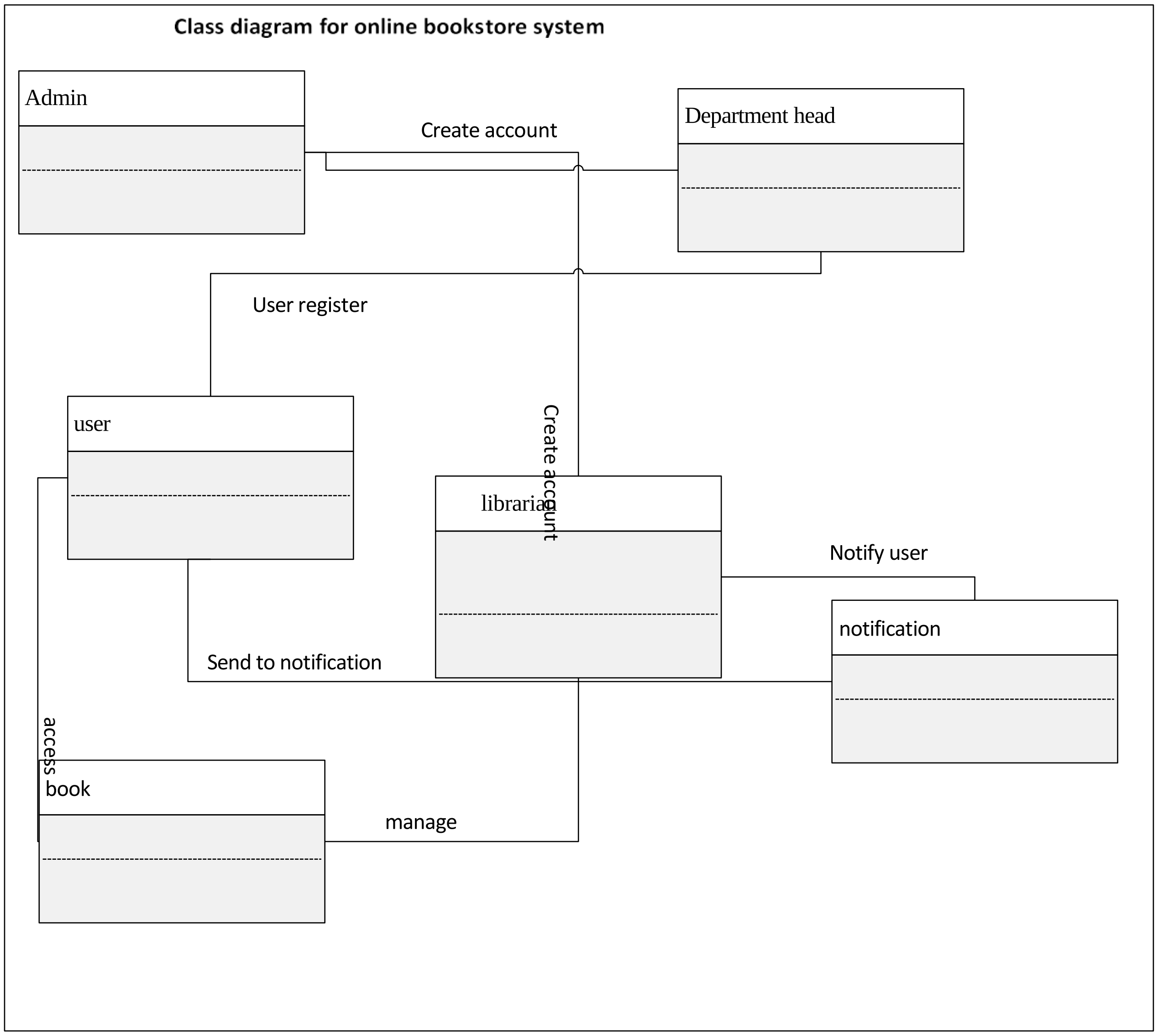


Figure 29: class diagram detail

5.2.3.1 Class Diagram Description

Class name	Description
Administrator	Represents the system administrator responsible for managing the system. The admin creates accounts for Department Heads and Librarians, ensuring they can access the system. Attributes include username, password, and name. Operations include creating accounts for Department Heads and Librarians and triggering notifications when new books are added by the Librarian.
User	Represents students or college employers who interact with the system to access books. Users can search for books, view book details, read books online, and download books. Attributes include username, password, and name. Operations include searching for books by title, author, or department, reading books online, downloading books, and receiving notifications about new books.
Librarian	Represents a Librarian responsible for managing the book inventory. The Librarian adds new books to the system and manages existing books through the Inventory Manager. Attributes include username, password, and name. Operations include storing books online (adding new books) and managing books (edit/delete). reading books online, downloading books, and receiving notifications about new books.
Book	Represents a book in the online bookstore, storing details for user access. Attributes include book, title, author, department, and availability. Operations include retrieving book details for display to Users and updating availability status when managed by the Librarian.
Notification.	Represents notifications generated by the system to inform Users about events, such as the addition of a new book. It is triggered by the Librarian's actions through the Inventory Manager. Attributes include notification, message, date created, and status (e.g., active, expired). Operations include generating a notification message when a new book is added and displaying it to Users on their main page.

Table 28: class diagram detail

5.2.4. Persistent Data Management

Persistent data management refers to the storage of the data that will be used in the system. The data that will be stored in the system include user data, book data. Persistent data management encompasses the way these data are stored, retrieved, and updated in the system.

Database Design: The system's database consists of several tables to manage different types of data:

- User Table: Stores user information such as username, password (hashed for security), and name.
- Book Table: Stores book details including title, author, department, and availability status (true/false).

Data Storage and Retrieval:

- MySQL tables are designed with appropriate primary and foreign keys to ensure data integrity and efficient retrieval.
- Data retrieval is optimized using SQL queries, with indexes applied to frequently searched fields like book title and author to improve performance.

Data Security:

- User passwords are hashed using PHP's password hash() function to protect sensitive information.
- Access to the database is restricted to the application server (via PHP scripts), preventing direct external access.

Performance Optimization: The performance of online bookstore system is critical to its success. The database should be optimized for fast access and retrieval of data. This includes the use of indexes, caching, and database partitioning.

5.2.5. Access Control and Security

Access control ensures that the system's functionalities are restricted based on user roles, allowing only authorized users to perform specific actions. The Web-Based Online Book Store System for ARU has two main user roles: Administrator and User (students or collage employer). The table below outlines the access privileges for each role

function	**student**	**administrator**	**librarian**
login	yes	yes	yes
User registration	no	yes	no
Create account	no	yes	no
Bookstore online	no	no	yes
Search book	yes	yes	yes
Read book online	yes	yes	yes
Download book	yes	yes	yes
Notification	no	yes	yes
Manage book	no	yes	yes
Manage user	no	yes	No
Book rate	yes	no	no
Send comment	Yes	no	no
Change password	yes	yes	yes
Approve comment	No	yes	yes
logout	yes	yes	yes

Table 29: for Access Control and Security

5.3. Packages

In order to develop the proposed system, we are going to use the following package that helps us to simplify development, enhance functionality, and maintain code efficiency of the system that we are going to develop.

PHP: is a server-side scripting language widely used for web development. It provides the core functionality for handling user requests, processing data, and interacting with the MySQL database in the Web-Based Online Book Store System.

MySQL: MySQL is an open-source relational database management system that provides packages for creating, managing, and querying databases. It is used to store and retrieve data such as user profiles, book records.

Bootstrap:

The Bootstrap framework provides a collection of CSS and JavaScript tools for building responsive and mobile-friendly web interfaces. It is used to design the user interface, including layouts for the home page, search interface, and admin panel, ensuring a consistent and modern look.

5.4. Algorithm Design

A system requires efficient algorithms to ensure that the system can handle large amounts of data and provide interactive use of the system for the user. Here are some key aspects of algorithm design.

- **Search Algorithm:** These algorithms can be used for fast retrieval of books based on specific criteria like title, author, or department. The system uses SQL queries to search the MySQL database efficiently.
- **Sorting Algorithm:** Sorting algorithms like quick-sort, merge sort, or others may be used for organizing books in the search results or inventory list, such as sorting by title or author for better user experience.
- Notification **Algorithms:** Efficient algorithms for sending timely notifications to users about due dates, new book additions request statuses through the web interface.

CHAPTER SIX

6.1 Implementation

The implementation phase of the Web-Based Online Book Store System for Arsi University involved translating the system design into a functional application. The system was developed using a three-tier architecture, comprising the client layer (frontend), application server layer (backend), and database layer. The implementation process focused on creating a user-friendly, secure, and efficient platform to enable students, librarians, and administrators to interact with academic resources seamlessly.

The system was built using the following technologies:

- **Frontend**: HTML, CSS, and JavaScript, with Bootstrap for responsive and mobile-friendly interfaces.
- **Backend**: PHP for handling business logic, user authentication, and book management.
- **Database**: MySQL for persistent data storage, managed through XAMPP.

The implementation followed the Agile methodology, allowing iterative development and continuous feedback. Key functionalities, such as user registration, book search, online reading, downloading, and notification generation, were developed in sprints, ensuring each module was tested and refined before integration.

6.2 Implementation of the Database

The database was implemented using MySQL, hosted on a local XAMPP server during development. The database schema was designed to support efficient storage and retrieval of user profiles, book records, notifications, and comments. The following tables were created to manage persistent data:

- **Administrator**: Stores admin credentials (username, password, name) for system management.
- **Librarian**: Stores librarian details (username, password, name) for book and user management.
- **User**: Stores student and college employer information (username, password, name) for authentication and access to books.
- **Book**: Stores book details (title, author, department, availability, file_path) to support search, reading, and downloading.

- **Notification**: Stores notification data (notification_id, message, date_created, status) for informing users about new books.
- **Comment**: Stores user-submitted comments (comment_id, book_id, user_id, content, status) for book feedback, pending approval.

6.1.1 Home page

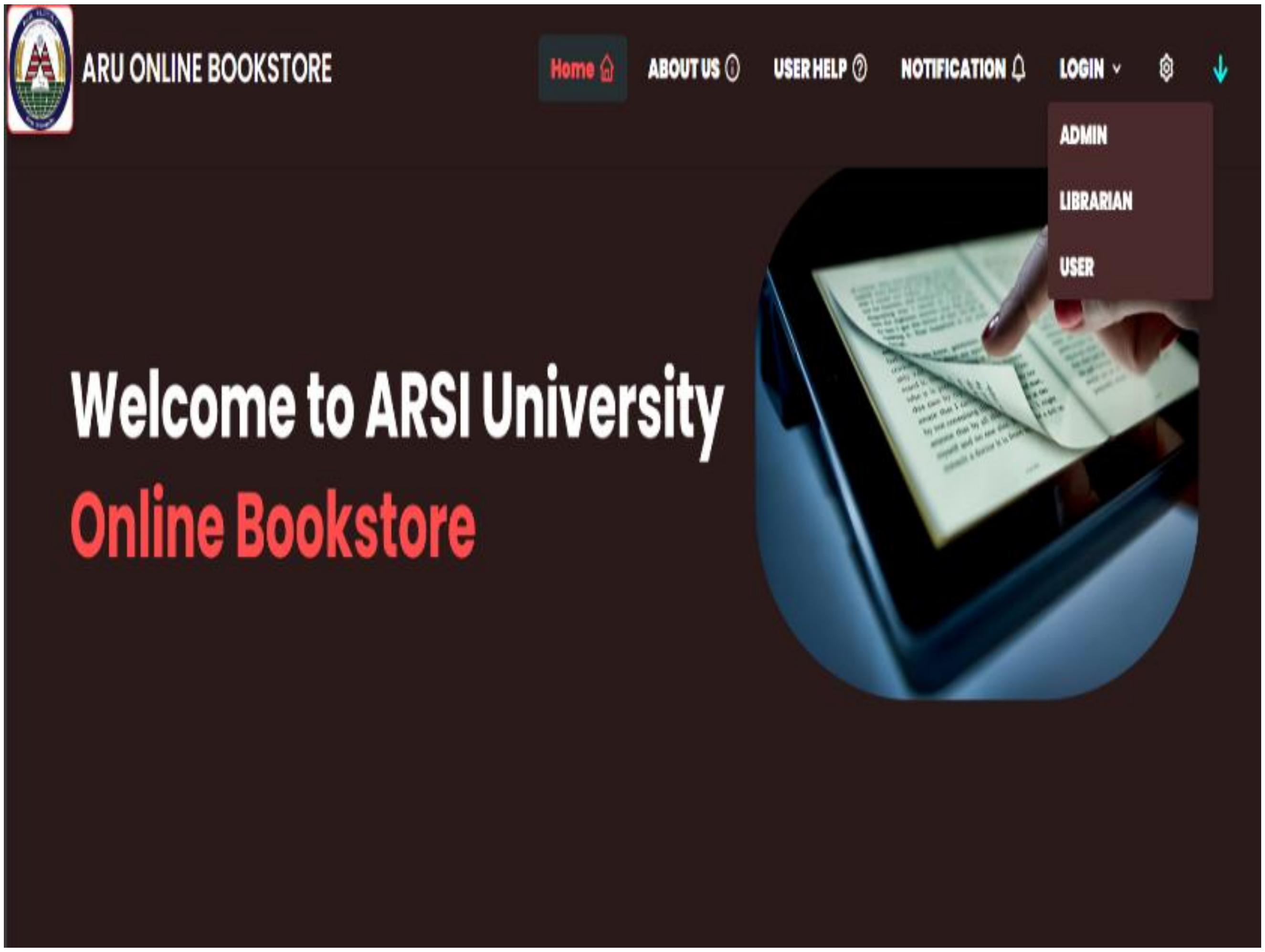


Figure 30:Home page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
    <title>ONLINE BOOKSTORE SYSTEM</title>
    <link
href='https://unpkg.com/boxicons@2.1.4/css/
boxicons.min.css' rel='stylesheet'>
    <script
src="https://unpkg.com/scrollreveal"></scri
pt>
    <script
src="https://cdn.jsdelivr.net/npm/typed.js@
2.0.12"></script>
    <link rel="stylesheet"
href="css/style.css">
</head>
<body>
    <?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "online_book_Db";

    function countBooks() {
        global $servername, $username,
$password, $dbname;
        $count = 0;
        $conn = new mysqli($servername,
$username, $password, $dbname);
        if ($conn->connect_error) {
            return 0;
        }
        $sql = "SELECT COUNT(file) as total
FROM books";
        $result = $conn->query($sql);
        if ($result && $result->num_rows >
0) {
            $row = $result->fetch_assoc();
            $count = $row['total'];
```

```
$password, $dbname);
        if ($conn->connect_error) {
            return 0;
        }
        $sql = "SELECT COUNT(*) as total FROM use
        $result = $conn->query($sql);
        if ($result && $result->num_rows > 0) {
            $row = $result->fetch_assoc();
            $count = $row['total'];
        }
        $conn->close();
        return $count;
    }

    function getActiveNotifications() {
        global $servername, $username, $password
        $notifications = array();
        $conn = new mysqli($servername, $usernam
        if ($conn->connect_error) {
            return $notifications;
        }
```

```
down'></i></a>
                <div class="dropdown-menu">
                    <a href="admin dashboard.php
                    <a href="librarian.php">LIBR
                    <a href="user.php">USER</a>
                </div>
            </div>
            <div class="theme-switcher">
                <a href="#setting" id="settings-
cog'></i></a>
                <div class="theme-options" id="t
                    <div class="theme-option" da
                    <div class="theme-option" da
                    <div class="theme-option" da
                    <div class="theme-option" da
                    <div class="theme-option" da
```

```
theme="light">Light</div>
                </div>
            </div>
            <a href="#footer" title="Go to Foote
                <i class='bx bx-down-arrow-alt'
aqua;"></i>
            </a>
```

```
        }
        $conn->close();
        return $count;
    }

    function countUsers() {
        global $servername, $username,
$password, $dbname;
        $count = 0;
        $conn = new mysqli($servername,
$username,

 <header class="header" id="header">
    <a href="" class="logo">
            <img src="image/logo.jpeg"
alt="Online Bookstore Logo" class="logo-
img">
            ARU ONLINE BOOKSTORE
        </a>
        <i class='bx bx-menu' id="menu-
icon"></i>
        <nav class="navbar">
            <a href="#home"
class="active">Home <i class='bx bx-
home'></i></a>
            <a href="#stats">ABOUT US <i
class='bx bx-info-circle'></i></a>
            <a href="#services">USER HELP
<i class='bx bx-help-circle'></i></a>
            <div
style="position:relative;display:inline-
block;">
                <a
href="notifications.php">NOTIFICATION <i
class='bx bx-bell'></i>
                    <span id="notification-
```

```
badge" class="notification-badge" style="<?php e
'display:none;' ?>"><?php echo $notificationCoun
                </a>
            </div>
            <div class="dropdown">
              <a href="#" class="dropdown-toggle
</nav>
    </header>
     <div class="footer-contact">
                <h3>Contact Us</h3>
                <p><i class='bx bx-mail-send'></
href="mailto:info@arsiuniversity.edu">pr.arsiu@a
                <p><i class='bx bx-phone'></i> <
href="tel:+1234567890">+251222380252</a></p>
            </div>
            <div class="footer-iconTop">
                <a href="#home" title="Back to T
                    <i class='bx bx-up-arrow-alt
                </a>
            </div>
        </div>
        <div class="footer-text">
            <p>Copyright ©2025 All Rights Reserv
        </div>
    </footer>
```

6.1.1 Administrator page

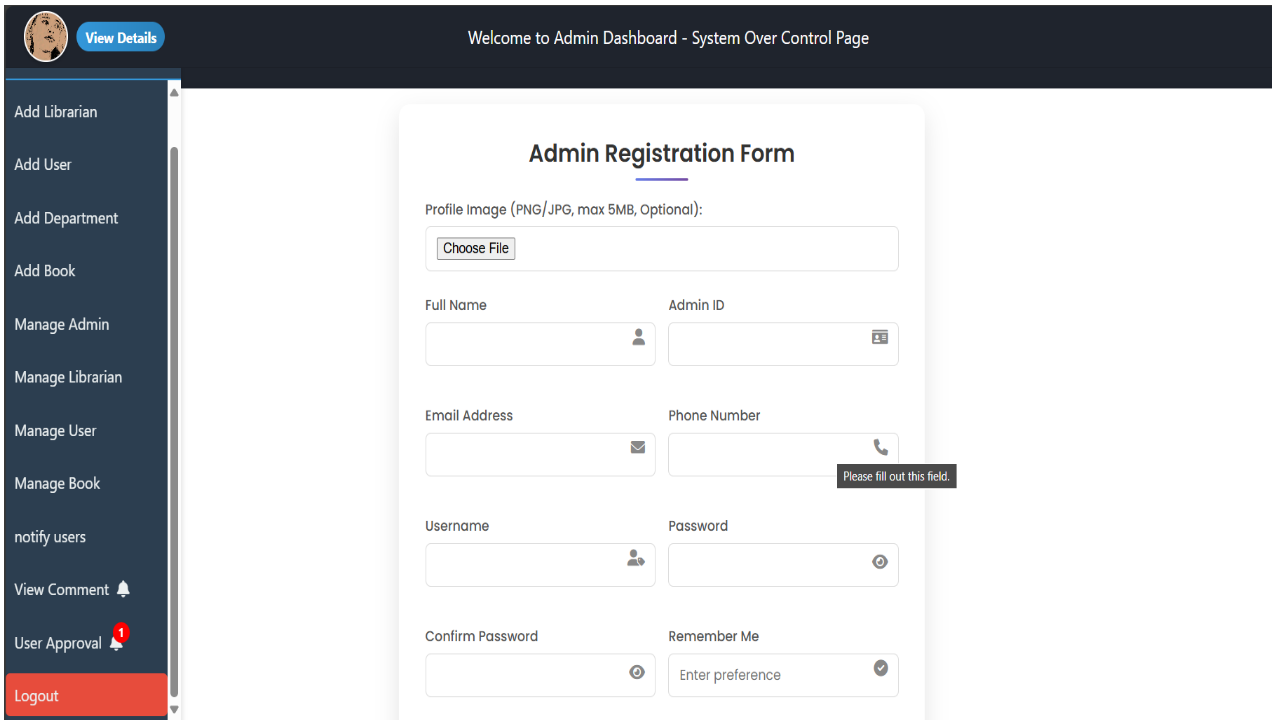


Figure 31:Administrator page

```php
<?php
session_start();

if (!isset($_SESSION['logged_in']) ||
!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
}

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "online_book_Db";

$conn = new mysqli($servername, $username,
$password, $dbname);
if ($conn->connect_error) {
    error_log("Connection failed: " . $conn-
>connect_error);
    die("Connection failed: " . $conn
->connect_error);
}

$admin_name =
isset($_SESSION['full_name']) ?
$_SESSION['full_name'] : "Admin
User";
$profile_image =
isset($_SESSION['profile_image']) ?
$_SESSION['profile_image'] : null;

$sql = "SELECT profile_image FROM
Admin WHERE id = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("i",
$_SESSION['user_id']);
$stmt->execute();
$result = $stmt->get_result();
if ($result->num_rows > 0) {
    $admin = $result->fetch_assoc();
```

```
        } else {
            $parts = explode(' ',
$accessPermission);
            if (count($parts) === 2 &&
is_numeric($parts[0])) {
                $duration = (int)$parts[0];
                $unit = $parts[1];
                $interval = ($unit === 'Week') ?
"weeks" : "months";
                $expirationDate = date('Y-m-d',
strtotime("+$duration $interval",
strtotime($startDate)));
            } else {
                continue;
            }
        }

        $remainingSeconds =
strtotime($expirationDate) -
strtotime($currentDate);
        $remainingDays = floor($remainingSeconds
/ (24 * 60 * 60));

        if ($remainingDays <= 30) {
            $approval_count++;
        }
    }
}
error_log("Dashboard - Approval count (30 days or
less): $approval_count");

$conn->close();
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>Admin Dashboard</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font
-awesome/6.0.0-beta3/css/all.min.css">
    <link rel="stylesheet" href="css/admin
```

```
    $profile_image =
$admin['profile_image'];
    $_SESSION['profile_image'] =
$profile_image;
}
$stmt->close();

$comment_count = 0;
$result = $conn->query("SELECT
COUNT(*) as count FROM comment");
if ($result && $result->num_rows >
0) {
    $row = $result->fetch_assoc();
    $comment_count = $row['count'];
}

$approval_count = 0;
$currentDate = date('Y-m-d');
$sql = "SELECT date,
access_permission FROM users";
$result = $conn->query($sql);

if ($result && $result->num_rows >
0) {
    while ($row = $result-
>fetch_assoc()) {
        $accessPermission =
$row['access_permission'];
        $startDate = $row['date'];

        if ($accessPermission ===
'Approved') {
            $expirationDate =
date('Y-m-d', strtotime("+30 days",
strtotime($startDate)));
dashboard.css">
    <link rel="stylesheet"
href="css/themes.css">
</head>
<body class="theme-switcher">
    <div class="header">
        <div class="profile-
container">
```

```
file_exists($_SERVER['DOCUMENT_ROOT'] .
"/bookstore/book/Admin/" . $profile_image)): ?>
                <img
src="/bookstore/book/Admin/<?php echo
htmlspecialchars($profile_image); ?>"
                     alt="Profile image of <?php
echo echo htmlspecialchars($admin_name); ?>"
                     class="profile-image">
            <?php else: ?>
                <div class="profile-image
initials">
                    <?php
                        $initials = '';
                        $names = explode(' ',
$admin_name);
                        foreach ($names as $n) {
                            $initials .=
strtoupper(substr($n, 0, 1));
                            if (strlen($initials)
>= 2) break;
                        }
                        echo
htmlspecialchars($initials);
                    ?>
                </div>
            <?php endif; ?>
            <a href="admin profile.php"
class="profile-button">View Details</a>
        </div>
        <span class="header-text">Welcome to
Admin Dashboard - System Over Control Page</span>
    </div>
    <div class="hamburger" aria-label="Show
menu">☰</div>
    <div class="container">
        <div class="sidebar active">
            <div class="menu">
                <a href="user/admin register
                    <span class="approval-count"
data-count="<?php echo $approval_count; ?>"
style="position: relative; display: inline-block;
margin-left: 5px;">
                        <i class="fas fa-bell
notification-icon"></i><?php if ($approval_count
```

```
                                    <?php   if
($profile_image &&
```

```
> 0): ?>
                            <span
class="notification-badge"
style="position: absolute; top: -
8px; right: -8px; background-color:
red; color: white; border-radius:
50%; padding: 2px 6px; font-size:
12px; font-weight: bold;">
                                <?ph
p echo $approval_count; ?>
                            </span>
                        <?php endif;
?>
                    </span>
                </a>
                <a href="logout.php"
class="menu-item logout">Logout</a>
            </div>
        </div>
        <div class="content-area">
            <iframe
id="contentFrame" class="content-
frame active" src="user/admin
register form.php"></iframe>
        </div>
    </div>
```

1.3 Librarian page

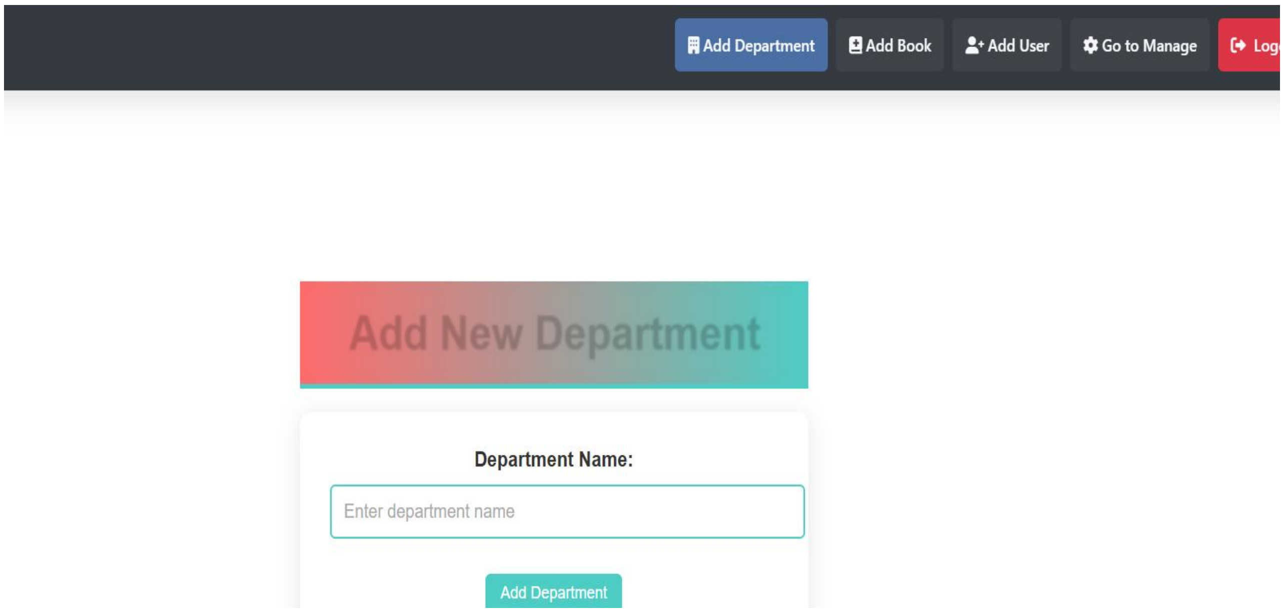
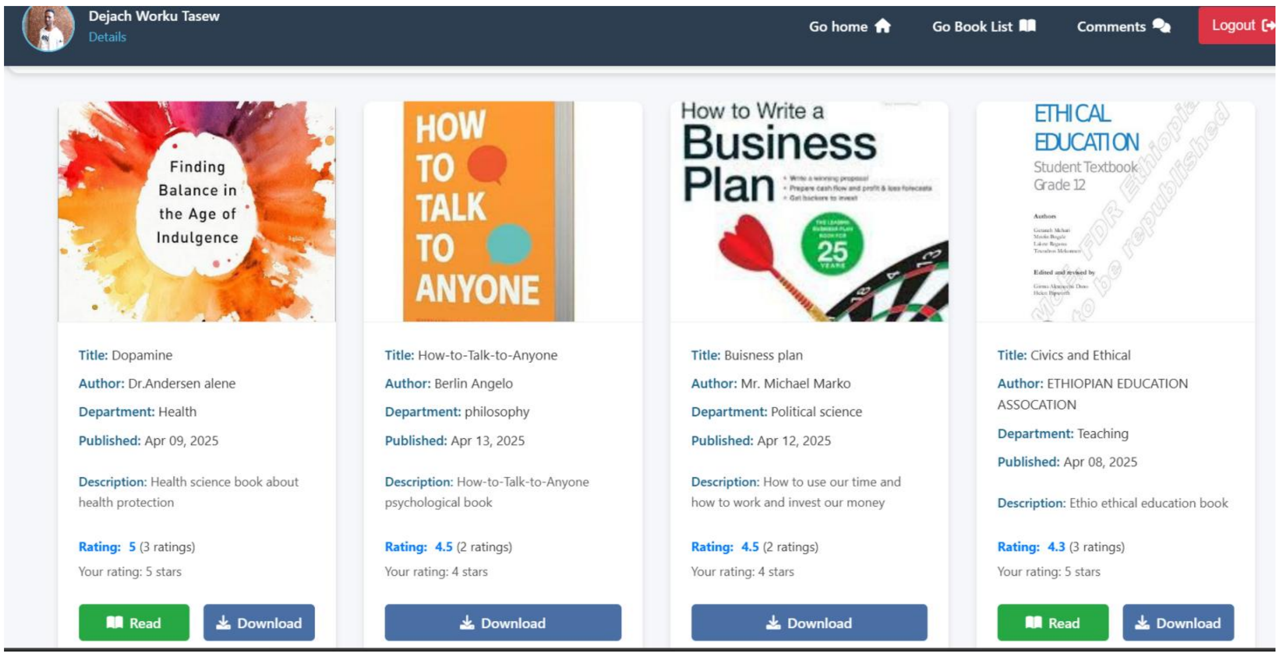


Figure 32:Librarian page

```
<?php
session_start();
if (!isset($_SESSION['logged_in']) ||
!isset($_SESSION['user_id']) ||
!isset($_SESSION['role']) ||
$_SESSION['role'] !== 'Librarian') {
    header("Location: login0.php");
    exit();}
$librarian_name =
isset($_SESSION['full_name']) &&
!empty($_SESSION['full_name']) ?
$_SESSION['full_name'] : "Librarian";
$profile_image =
isset($_SESSION['profile_image']) &&
!empty($_SESSION['profile_image']) ?
$_SESSION['profile_image'] : null;
$base_path = '/bookstore/book/Librarian/';
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
      echo htmlspecialchars($initials);
  ?> </div>
 <?php endif; ?><a href="librarian
profile.php" class="view-details">View
Details</a</div><nav class="nav-menu">
            <a href="#" data-
page="dep't.php"><i class="fas fa-
building"></i> Add Department</a>
            <a href="#" data-page="add
book.php"><i class="fas fa-book-
medical"></i> Add Book</a>
            <a href="#" data-
page="user/user register form.php"><i
class="fas fa-user-plus"></i> Add User</a>
            <a href="menu.php"><i
class="fas fa-cog"></i> Go to Manage<
<button class="logout"><i class="fas fa-
sign-out-alt"></i>
Logout</button>      </div>
```

6.1.4 User page


Dejach Worku Tasew
Details
Go home | Go Book List | Comments | Logout

Title: Dopamine
Author: Dr.Andersen alene
Department: Health
Published: Apr 09, 2025
Description: Health science book about health protection
Rating: 5 (3 ratings)
Your rating: 5 stars
Read | Download

Title: How-to-Talk-to-Anyone
Author: Berlin Angelo
Department: philosophy
Published: Apr 13, 2025
Description: How-to-Talk-to-Anyone psychological book
Rating: 4.5 (2 ratings)
Your rating: 4 stars
Download

Title: Buisness plan
Author: Mr. Michael Marko
Department: Political science
Published: Apr 12, 2025
Description: How to use our time and how to work and invest our money
Rating: 4.5 (2 ratings)
Your rating: 4 stars
Download

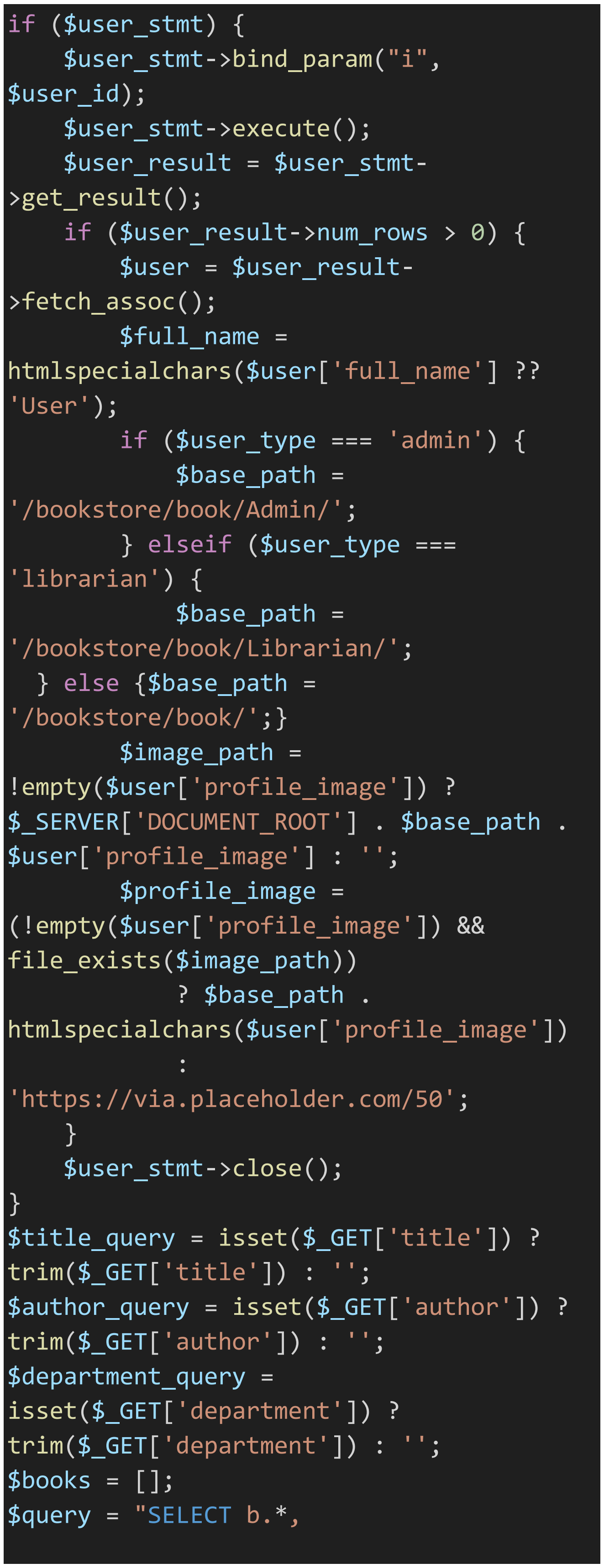
Title: Civics and Ethical
Author: ETHIOPIAN EDUCATION ASSOCATION
Department: Teaching
Published: Apr 08, 2025
Description: Ethio ethical education book
Rating: 4.3 (3 ratings)
Your rating: 5 stars
Read | Download

Finding Balance in the Age of Indulgence
HOW TO TALK TO ANYONE
How to Write a Business Plan
25
ETHICAL EDUCATION
Student Textbook Grade 12

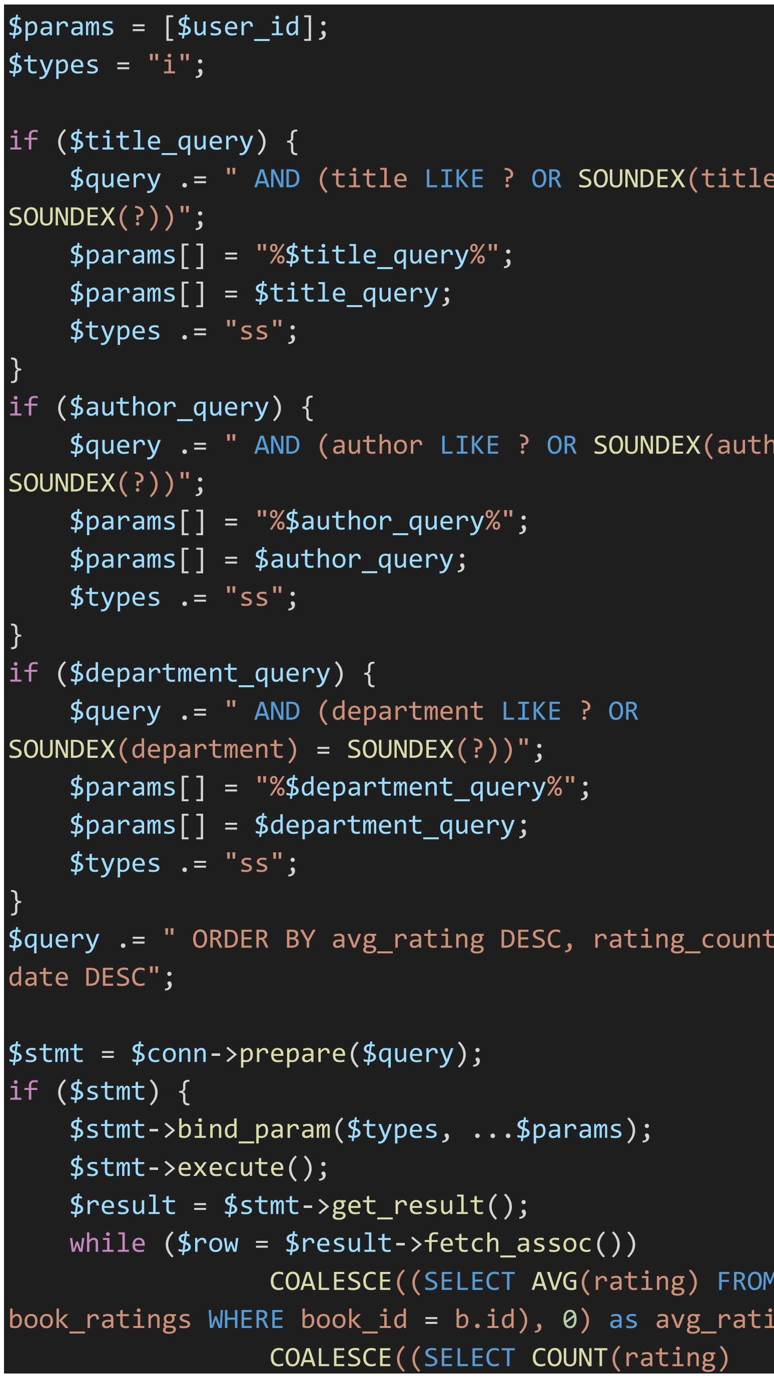
Figure 33:user page

```php
<?php
session_start();

if (!isset($_SESSION['logged_in']) ||
$_SESSION['logged_in'] !== true) {
    header("Location: login1.php");
    exit;
}

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "online_book_Db";

$conn = new mysqli($servername,
$username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}
```

```php
$user_id = $_SESSION['user_id'];
$user_type = isset($_SESSION['user_type']) ?
$_SESSION['user_type'] : 'user';

if ($user_type === 'user') {
    $user_stmt = $conn->prepare("SELECT full_name,
profile_image FROM users WHERE id = ?");
} elseif ($user_type === 'admin') {
    $user_stmt = $conn->prepare("SELECT full_name,
profile_image FROM admin WHERE id = ?");
} elseif ($user_type === 'librarian') {
    $user_stmt = $conn->prepare("SELECT full_name,
profile_image FROM librarian WHERE id = ?");
} else {
    $user_stmt = false;
}
$full_name = 'User'; $profile_image =
'https://via.placeholder.com/50';
FROM book_ratings WHERE book_id = b.id), 0) as rat
                 (SELECT rating FROM book_ratings
book_id = b.id AND user_id = ?) as user_rating
          FROM books b WHERE 1=1";
```

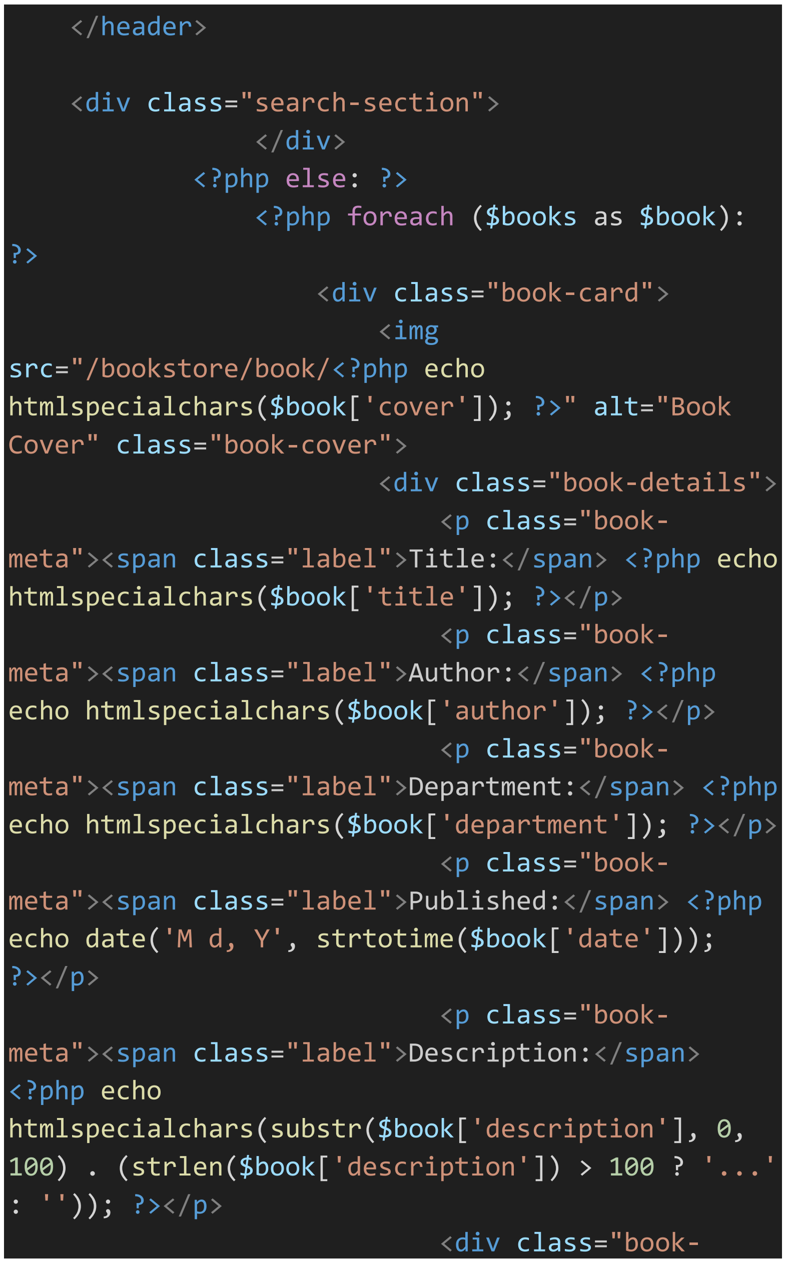


```
if ($user_stmt) {
    $user_stmt->bind_param("i",
$user_id);
    $user_stmt->execute();
    $user_result = $user_stmt-
>get_result();
    if ($user_result->num_rows > 0) {
        $user = $user_result-
>fetch_assoc();
        $full_name =
htmlspecialchars($user['full_name'] ??
'User');
        if ($user_type === 'admin') {
            $base_path =
'/bookstore/book/Admin/';
        } elseif ($user_type ===
'librarian') {
            $base_path =
'/bookstore/book/Librarian/';
  } else {$base_path =
'/bookstore/book/';}
        $image_path =
!empty($user['profile_image']) ?
$_SERVER['DOCUMENT_ROOT'] . $base_path .
$user['profile_image'] : '';
        $profile_image =
(!empty($user['profile_image']) &&
file_exists($image_path))
            ? $base_path .
htmlspecialchars($user['profile_image'])
            :
'https://via.placeholder.com/50';
    }
    $user_stmt->close();
}
$title_query = isset($_GET['title']) ?
trim($_GET['title']) : '';
$author_query = isset($_GET['author']) ?
trim($_GET['author']) : '';
$department_query =
isset($_GET['department']) ?
trim($_GET['department']) : '';
$books = [];
$query = "SELECT b.*,
```

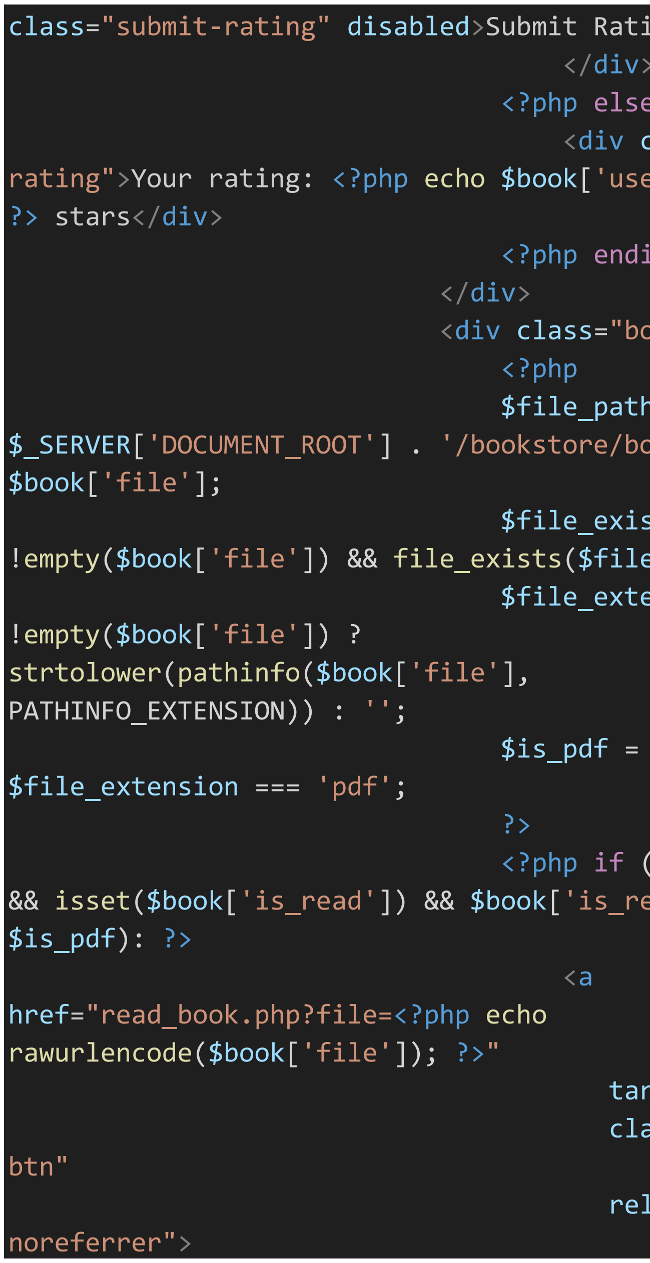


```
$params = [$user_id];
$types = "i";

if ($title_query) {
    $query .= " AND (title LIKE ? OR SOUNDEX(title
SOUNDEX(?))";
    $params[] = "%$title_query%";
    $params[] = $title_query;
    $types .= "ss";
}
if ($author_query) {
    $query .= " AND (author LIKE ? OR SOUNDEX(auth
SOUNDEX(?))";
    $params[] = "%$author_query%";
    $params[] = $author_query;
    $types .= "ss";
}
if ($department_query) {
    $query .= " AND (department LIKE ? OR
SOUNDEX(department) = SOUNDEX(?))";
    $params[] = "%$department_query%";
    $params[] = $department_query;
    $types .= "ss";
}
$query .= " ORDER BY avg_rating DESC, rating_count
date DESC";

$stmt = $conn->prepare($query);
if ($stmt) {
    $stmt->bind_param($types, ...$params);
    $stmt->execute();
    $result = $stmt->get_result();
    while ($row = $result->fetch_assoc())
                 COALESCE((SELECT AVG(rating) FROM
book_ratings WHERE book_id = b.id), 0) as avg_rati
                 COALESCE((SELECT COUNT(rating)
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>User Page</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0-beta3/css/all.min.css">
    <link rel="stylesheet"
href="css/userpage.css">
    <link rel="stylesheet" href="css/themes.css">
</head>
<body>
    <header id="header">
        <div class="header-content">
            <div class="logo-container">
                <img src="<?php echo
$profile_image; ?>" alt="User Profile">
            </div>
            <div class="user-info">
                <span><?php echo $full_name;
?></span>
                <?php if ($user_type === 'user'):
?>
                    <a href="user
profile.php">Details</a>
                <?php endif; ?>
            </div>
        </div>
        <div class="hamburger">
            <i class="fas fa-bars"></i>
        </div>
        <nav class="nav-menu">
            <a href="index.php">Go home <i
class="fas fa-home"></i></a>
            <a href="view_book_list.php">Go Book
List <i class="fas fa-book-open"></i></a>
            <a href="comment.php">Comments <i
class="fas fa-comments"></i></a>
            <button class="logout">Logout <i
class="fas fa-sign-out-alt"></i></button>
        </nav>
        <div class="overlay"></div>
```

```
     <div class="search-container">
            <div class="search-input-conta
                <label for="title-input"
class="search-label">Search by:</label>
                <input type="text" name="t
id="title-input" class="search-input"
placeholder="Search by title" value="<?php
htmlspecialchars($title_query); ?>">
                <input type="text" name="a
id="author-input" class="search-input"
placeholder="Search by author" value="<?ph
htmlspecialchars($author_query); ?>">
                <input type="text" name="d
id="department-input" class="search-input"
placeholder="Search by department" value="
htmlspecialchars($department_query); ?>">
                <button type="submit" clas
icon"><i class="fas fa-search"></i></butto
                <button type="button" clas
search" title="Clear Search"><i class="fas
alt"></i></button>
            </div>
        </div>
    </div>

    <main id="book-list">
        <div class="book-grid">
            <?php if (empty($books) && ($t
|| $author_query || $department_query)): ?
                <div class="no-books">
                    <i class="fas fa-book-
3x"></i>
                    <h3>No Book Found</h3>
                    <p>No book matches you
criteria.</p>
                </div>
            <?php elseif (empty($books) &&
!($title_query || $author_query ||
$department_query)): ?>
                <div class="no-books">
                    <i class="fas fa-book-
3x"></i>
                    <h3>No Books Available
                    <p>There are currently
the database.</p>
```



```
    </header>

    <div class="search-section">
                </div>
            <?php else: ?>
                <?php foreach ($books as $book):
?>
                    <div class="book-card">
                        <img
src="/bookstore/book/<?php echo
htmlspecialchars($book['cover']); ?>" alt="Book
Cover" class="book-cover">
                        <div class="book-details">
                            <p class="book-
meta"><span class="label">Title:</span> <?php echo
htmlspecialchars($book['title']); ?></p>
                            <p class="book-
meta"><span class="label">Author:</span> <?php
echo htmlspecialchars($book['author']); ?></p>
                            <p class="book-
meta"><span class="label">Department:</span> <?php
echo htmlspecialchars($book['department']); ?></p>
                            <p class="book-
meta"><span class="label">Published:</span> <?php
echo date('M d, Y', strtotime($book['date']));
?></p>
                            <p class="book-
meta"><span class="label">Description:</span>
<?php echo
htmlspecialchars(substr($book['description'], 0,
100) . (strlen($book['description']) > 100 ? '...'
: '')); ?></p>
                            <div class="book-
```



```
class="submit-rating" disabled>Submit Rati
                                    </div
                                <?php else
                                    <div c
rating">Your rating: <?php echo $book['use
?> stars</div>
                                <?php endi
                            </div>
                            <div class="bo
                                <?php
                                $file_path
$_SERVER['DOCUMENT_ROOT'] . '/bookstore/bo
$book['file'];
                                $file_exis
!empty($book['file']) && file_exists($file
                                $file_exte
!empty($book['file']) ?
strtolower(pathinfo($book['file'],
PATHINFO_EXTENSION)) : '';
                                $is_pdf =
$file_extension === 'pdf';
                                ?>
                                <?php if (
&& isset($book['is_read']) && $book['is_re
$is_pdf): ?>
                                    <a
href="read_book.php?file=<?php echo
rawurlencode($book['file']); ?>"
                                       tar
                                       cla
btn"
                                       rel
noreferrer">
```

6.2 Implementation of the Database

The database was implemented using MySQL, hosted on a local XAMPP server during development. The database schema was designed to support efficient storage and retrieval of user profiles, book records, notifications, and comments. The following tables were created to manage persistent data:

- **Administrator:** Stores admin credentials (username, password, name) for system management.

```sql
CREATE TABLE `admin` (
  `id` int(11) NOT NULL,
  `full_name` varchar(100) NOT NULL,
  `admin_id` varchar(50) NOT NULL,
  `email` varchar(100) NOT NULL,
  `phone` varchar(20) NOT NULL,
  `username` varchar(100) NOT NULL,
  `password` varchar(100) NOT NULL,
  `remember_me` text NOT NULL,
  `profile_image` varchar(255) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- **Librarian:** Stores librarian details (username, password, name) for book and user management.

```sql
CREATE TABLE `librarian` (
  `id` int(11) NOT NULL,
  `full_name` varchar(50) NOT NULL,
  `personal_id` varchar(50) NOT NULL,
  `email` varchar(100) NOT NULL,
  `phone` varchar(20) NOT NULL,
  `username` varchar(100) NOT NULL,
  `password` varchar(100) NOT NULL,
  `remember_me` text NOT NULL,
  `profile_image` varchar(255) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- **User**: Stores student and college employer information (username, password, name) for authentication and access to books.

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `date` date NOT NULL,
  `academic_year` varchar(4) NOT NULL,
  `full_name` varchar(100) NOT NULL,
  `id_number` varchar(50) NOT NULL,
  `department` varchar(100) NOT NULL,
  `year` enum('1st','2nd','3rd','4th','5th','6th','7th') NOT NULL,
  `semester` enum('1st','2nd') NOT NULL,
  `phone` varchar(10) NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(255) NOT NULL,
  `profile_image` varchar(255) NOT NULL,
  `remember_me` varchar(100) NOT NULL,
  `access_permission` varchar(20) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- **Notification:** Stores notification data (notification_id, message, date_created, status) for informing users about new books.

```
CREATE TABLE `notifications` (

  `id` int(11) NOT NULL,

  `book_id` int(11) NOT NULL,

  `availability` varchar(10) NOT NULL,

  `created_at` datetime DEFAULT current_timestamp(),

  `expiry_date` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- **Comment:** Stores user-submitted comments (comment_id, book_id, user_id, content, status) for book feedback, pending approval.

```
CREATE TABLE `comment` (

  `id` int(11) NOT NULL,

  `full_name` varchar(100) NOT NULL,
  `username` varchar(100) NOT NULL,
  `department` varchar(100) NOT NULL,
  `subject` varchar(150) NOT NULL,
  `message` text NOT NULL,
  `date` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

6.3 Implementation of Class Diagram

The class diagram outlined in Chapter Four was implemented using PHP classes to represent the system's core entities: Administrator, Librarian, User, Book, and Notification. Each class encapsulated attributes and operations as defined in the design phase.

- **Administrator Class**: Manages system-wide operations, such as creating librarian accounts and triggering notifications. Implemented with methods like add Librarian () and send Notification ().
- **Librarian Class**: Handles book inventory and user management. Includes methods like add Book (), edit Book (), and delete Book () to manage the book catalog.
- **User Class**: Represents students and college employers, with methods like search Book (), read Book(), and download Book() to interact with books.
- **Book Class**: Stores book metadata and supports operations like get Details () and update Availability() for displaying and managing book information.
- **Notification Class**: Manages system notifications, with methods like generate Notification ()
- **Views**: HTML templates styled with CSS and Bootstrap, dynamically populated with data using PHP.
- **Controllers**: PHP scripts handling user requests, coordinating between models and views, and managing session data.

6.4 Configuration of Application Server

The application server was configured using Apache, bundled with XAMPP, to host the web-based system. The configuration process included:

- **XAMPP Installation**: XAMPP was installed to provide Apache, MySQL, and PHP in a single package. The XAMPP control panel was used to start and stop services.

- **PHP Configuration**: The php.ini file was updated to enable PDO extensions for MySQL and increase the maximum file upload size (upload_max_filesize = 20M) to accommodate book file uploads (e.g., PDFs).

6.5 Configuration of Application Security

Security was a critical focus during implementation to protect user data and system integrity. The following measures were implemented:

- **User Authentication**: Login functionality was secured using PHP sessions and password hashing. Users must enter a valid username and password, validated against hashed credentials in the database. Session timeouts were set to 30 minutes to prevent unauthorized access.
- **Role-Based Access Control**: Access was restricted based on user roles. For example, only administrators could access the admin_dashboard.php page to manage librarians, while students were limited to user_dashboard.php for browsing and downloading books.
- **Data Encryption**: Sensitive data, such as passwords, were hashed using PHP's password_hash() function..
- **Input Validation**: All user inputs (e.g., search queries, registration forms) were sanitized using PHP's filter_var() and MySQL prepared statements to prevent SQL injection and cross-site scripting (XSS) attacks.
- **Secure File Uploads**: Book file uploads by librarians were restricted to PDF format, with size limits (20 MB) and virus scanning using a server-side script to ensure safe storage.

Security configurations were tested to ensure compliance with the non-functional requirements outlined in Chapter Three, such as preventing unauthorized access and protecting user privacy.

6.6 Implementation of User Interface

The user interface was implemented to provide an intuitive and responsive experience for all users. The frontend was developed using HTML, CSS, JavaScript, and Bootstrap, ensuring compatibility across devices (desktops, tablets, smartphones). Key interface components included:

- **Home Page**: Displays a search bar, featured books, and navigation links to login, register, and browse books. Styled with Bootstrap for a modern, grid-based layout.
- **Login Page**: A form for users to enter their username and password, with error messages for invalid credentials. Includes a “Forgot Password” link (implemented as a placeholder for future functionality).
- **User menu**: Allows students to search books by title, author, or department, view book details, read books online, and download files. Includes a notification panel for new book alerts.
- **Admin Dashboard**: Provides administrators with tools to manage librarians, users, and books. Features tabular views for editing/deleting records and forms for adding new entities.
- **Librarian menu**: Enables librarians to add, edit, or delete books and approve user comments. Includes a file upload form for adding book PDFs.

6.7 Testing

Testing was conducted to ensure the system met its functional and non-functional requirements, was free of defects, and provided a reliable user experience. The testing methodology included unit testing, integration testing, system testing, and acceptance testing, as outlined below.

6.7.1 Testing Tools and Environment

1. **Testing Tools**:

- ✓ **PHP Unit**: Used for unit testing PHP functions and classes, such as book search and user authentication logic.
- ✓ **MySQL Workbench**: Executed SQL queries to verify database integrity and performance.
- ✓ **Browser Developer Tools**: Debugged frontend issues (e.g., CSS layout, JavaScript errors) in Chrome and Firefox.

2. **Testing Environment**:

- ✓ **Server**: Local XAMPP server on indow 10-11 with Apache, MySQL, and PHP 8.2.

6.7.2 Unit Testing

Unit tests were conducted on individual components to verify their correctness in isolation. PHPUnit scripts were written for key functions, including:

- **Login Function**: Tested valid and invalid credential scenarios, ensuring correct redirects to role-specific dashboards and error messages.
- **Search Function**: Verified that searches by title, author, or department returned accurate results and handled empty queries appropriately.
- **Book Upload Function**: Ensured only PDF files under 50 MB were accepted and stored correctly in the database and server.
- **Notification Generation**: Confirmed that adding a new book triggered a notification stored in the database with the correct message.

6.7.3 Integration Testing

Integration testing verified that components worked together as expected. Test cases focused on interactions between the frontend, backend, and database, including:

- **User Registration and Login**: Ensured a newly registered user could log in and access the user dashboard without errors.
- **Book Search and Download**: Confirmed that searching for a book displayed results, and clicking "Download" saved the correct PDF file.
- **Notification Delivery**: Verified that adding a book by a librarian generated a notification visible on the user dashboard upon login.

Integration tests identified and resolved issues, such as a database connection timeout during high user load, by optimizing MySQL connection pooling.

6.7.4 System Testing

System testing evaluated the entire system against functional and non-functional requirements. Test scenarios included:

- **End-to-End User Flow**: A student registers, logs in, searches for a book, reads it online, downloads it, and rates it. The system performed as expected, with pages loading in under 2 seconds.
- **Performance Testing**: Simulated 500 concurrent users using Selenium scripts, confirming the server-maintained response times under 3 seconds.
- **Security Testing**: Attempted SQL injection and XSS attacks, verifying that input sanitization and prepared statements prevented vulnerabilities.

System testing confirmed that the system met requirements for usability, performance, and security, with minor UI adjustments made for better mobile responsiveness.

6.7.5 Acceptance Testing

Acceptance testing involved real users, including 10 ARU students, 2 librarians, and 1 administrator, to validate the system's usability and functionality. Users performed tasks such as:

- ✓ Registering and logging in.
- ✓ Searching for and downloading books.
- ✓ Adding books and approving comments (librarians and administrators only).

Feedback was collected via surveys, highlighting positive aspects (e.g., "easy to search and download books") and minor issues (e.g., "notification text could be clearer"). All critical functionalities were accepted, with suggestions for future enhancements, such as adding a book recommendation feature.

CHAPTER SEVEN

7. CONCLUSION AND RECOMMENDATION

7.1 Conclusion

The development of the Web-Based Online Book Store System for Arsi University successfully addressed the limitations of the existing physical bookstore system, providing a modern, efficient, and accessible platform for students, librarians, and administrators. The project aimed to digitize access to academic resources, enabling users to search, read, and download books seamlessly while improving inventory management and user experience. By leveraging web technologies such as PHP, MySQL, HTML, CSS, and JavaScript, the system was implemented as a robust, user-friendly solution tailored to the needs of the university community.

Key objectives, including requirement gathering, system analysis, design, implementation, and testing, were met through an Agile development approach. The system's three-tier architecture ensured scalability and maintainability, while features like role-based access control, real-time notifications, and secure data management enhanced its functionality and security. Testing phases, including unit, integration, system, and acceptance testing, confirmed that the system met both functional and non-functional requirements, delivering a reliable and intuitive experience.

The project significantly improved access to educational resources by eliminating the constraints of physical bookstore hours and manual processes. Students can now access up-to-date books anytime, anywhere, while librarians benefit from streamlined book management. The system also reduced operational costs and environmental impact by minimizing paperwork. Overall, the Web-Based Online Book Store System represents a transformative step toward modernizing resource access at Arsi University, supporting its academic mission and fostering a more efficient learning environment.

7.2 Recommendation

While the Web-Based Online Book Store System successfully meets its current objectives, there are opportunities for further enhancements to improve its functionality, user engagement, and long-term sustainability. The following recommendations are proposed for future development:

1. **Book Recommendation Engine**: Implement a machine learning-based recommendation system to suggest books based on users' search history, department, or ratings. This would enhance user engagement and help students discover relevant resources more easily.
2. **Mobile Application**: Develop a dedicated mobile app for iOS and Android to complement the web platform, providing offline access to downloaded books and push notifications for new book additions, improving accessibility for students on the go.
3. **Multi-Language Support**: Add support for multiple languages, such as Amharic and Oromo, to make the platform more inclusive for students from diverse linguistic backgrounds.

Reference

https://www.slideshare.net/computerized grading system.

ARSI University student library

https://paperswithcode.com/

(PDF) Online Bookstore—A New Trend in Textbook Sales Management for Services Marketing. (n.d.).

ResearchGate. Retrieved February 27, 2025, fromhttps://www.researchgate.net/publication/320708623_Online_Bookstore_-

A_New_Trend_in_Textbook_Sales_Management_for_Services_Marketing

The application of data mining in online bookstore | Request PDF. (n.d.). Retrieved February 27, 2025, from https://www.researchgate.net/publication/221544371_The_application_of_data_mining_in_onlin e_bookstore

Addis Ababa university E-book system

Haleem, A., Javaid, M., Qadri, M. A., & Suman, R. (2022). Understanding the role of digital technologies in education: A review. *Sustainable Operations and Computers*, *3*, 275–285. https://doi.org/10.1016/j.susoc.2022.05.004