Gollis University

Faculty of Software Engineering

Title:

Digital Library Management System

Submitted by:

1.Sakariye Qaasim Sh Omer

2.Maxamed Jaamac Abdiraxmaan

3.Aydaruus Maxamed Yousuf

4.Shacayb

Submitted to: Eng Hamse Suleyman

Submitted date: 29/October /2025th

Table of Contents

1.0 Introduction

This document provides a comprehensive Software Requirements Specification (SRS) for the Digital Library System. The goal of this project is to design and implement a platform that provides students and faculty member's easy access to digital educational materials. The system allows users to search, and read materials such as books, articles, and research papers electronically.

1.1 Purpose

The purpose of this project is to replace the traditional manual library system with a digital, automated one. By providing access to online resources, students can study anywhere, anytime. It aims to improve the efficiency of information retrieval, reduce time spent on manual searches, and provide a modern platform for knowledge sharing within the university.

1.2 Project Proposal

The Digital Library System will act as a central repository for educational materials. Users can create accounts, browse available materials, and download or read them online. Administrators can upload new materials, manage user accounts, and monitor system performance. This system promotes knowledge accessibility, reduces paper usage, and supports sustainable learning.

2.0 Overall Description

2.1 System Environment

The system will operate on a client-server architecture. The backend server will handle data processing and storage, while the frontend interface will allow users to interact with the system through a web browser or mobile device. The system will be accessible 24/7 and compatible with different devices and operating systems.

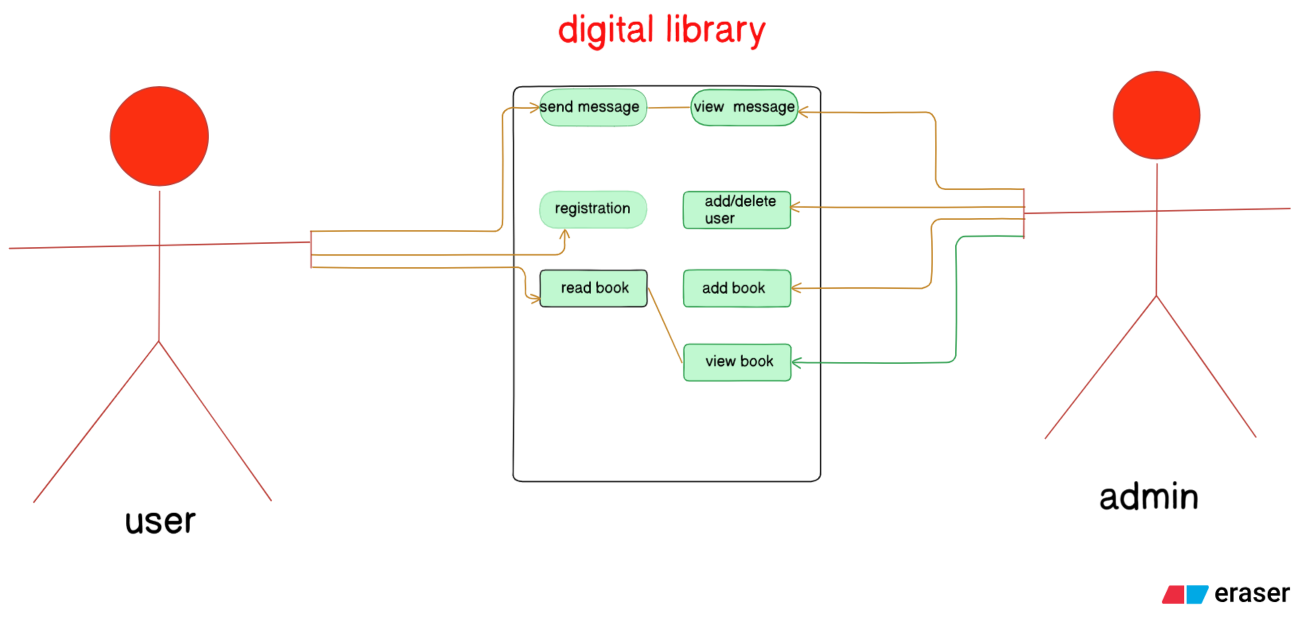


Figure 1 - System Environment

2.2 Functional Requirements Specification

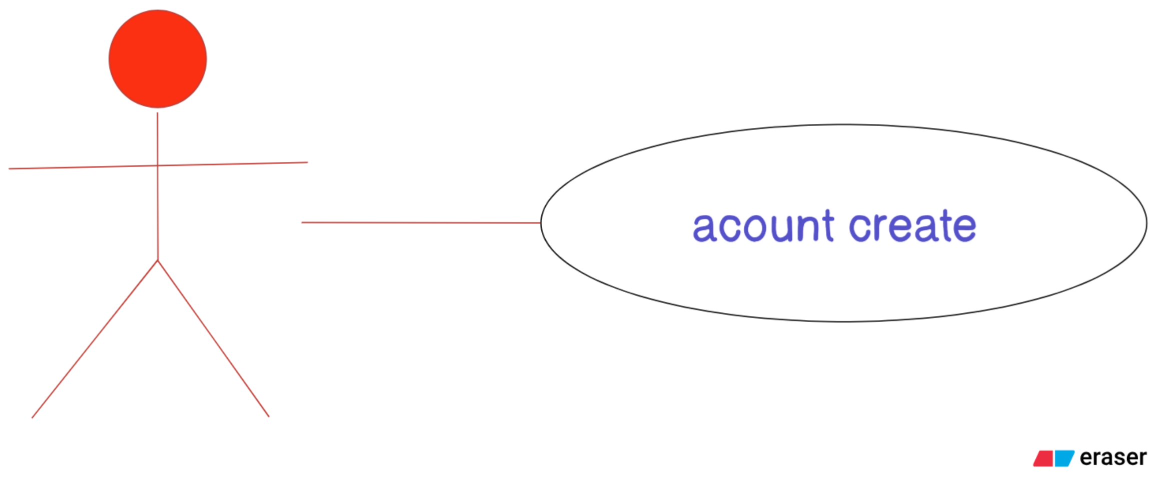
This section outlines the use cases for each of the activities in the program.

2.2.1 Register users

Use Case Use case: Account Create

Actor: user

w



Brief Description:

If the user wants to use the digital library, they must register within the system to gain access to its resources and services.

Initial Step-By-Step Description

Before this use case can be initiated, the user has already the program

1) The user selects the must to create an Account**.**
2) The system displays a registration form requiring details such as name, email, password, and user type (e.g., admin user).
3) The user fills in the required information and submits the form.
4) After successful login, the user can access the digital library and utilize available features (e.g., searching, reading or downloading digital books).

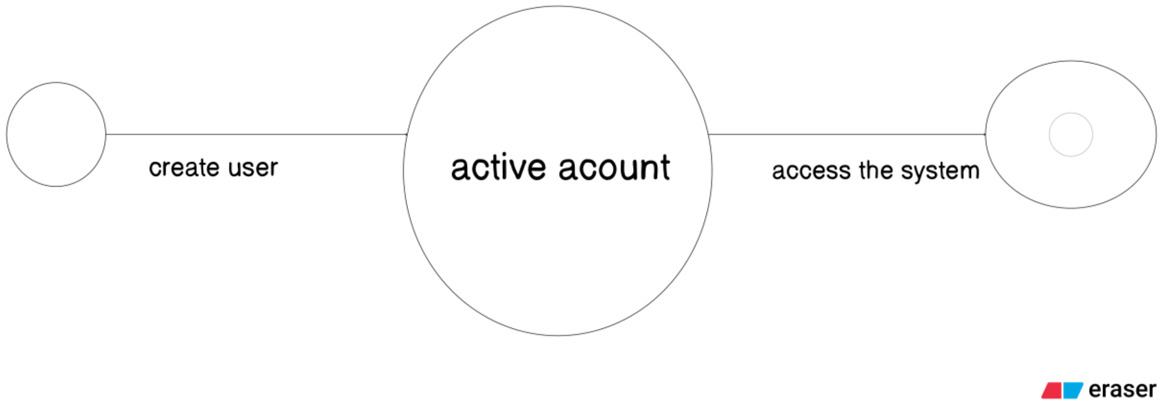


Figure 2 - Create account process

2.2.2 User Use Case

The User after register has the following:

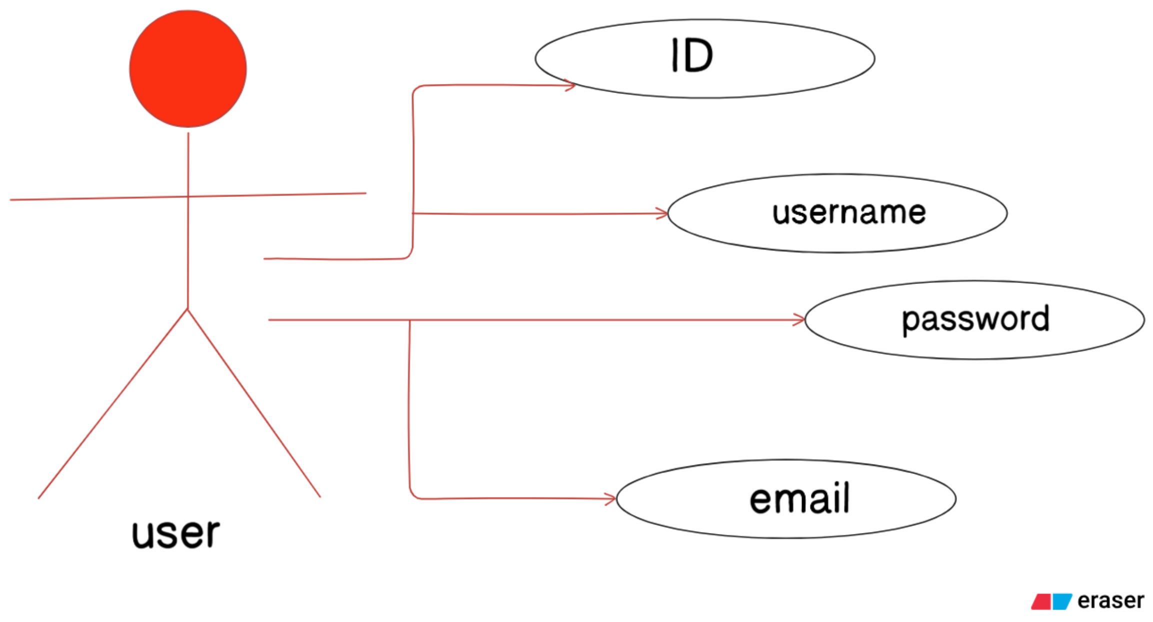


Figure 3 - User Use Cases

2.2.3 Use Case: User reading & download

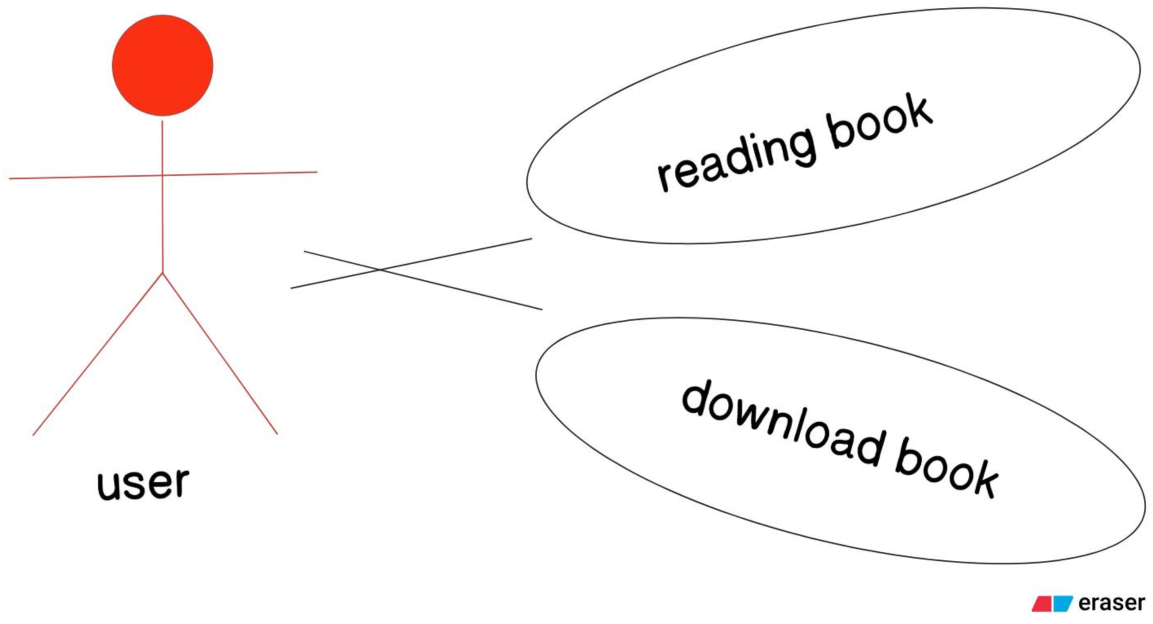


Figure 4 – User reading & download book

Description

Before this use case can be initiated, the user has accessed to the main page of the application.

The user can be read and download the book

2.2.4 Help ER Diagram

In case of helping ask, this term refers to the way for using the program.

Entity Relationship: Help asking Diagram:

Figure 5 - Help asking Diagram

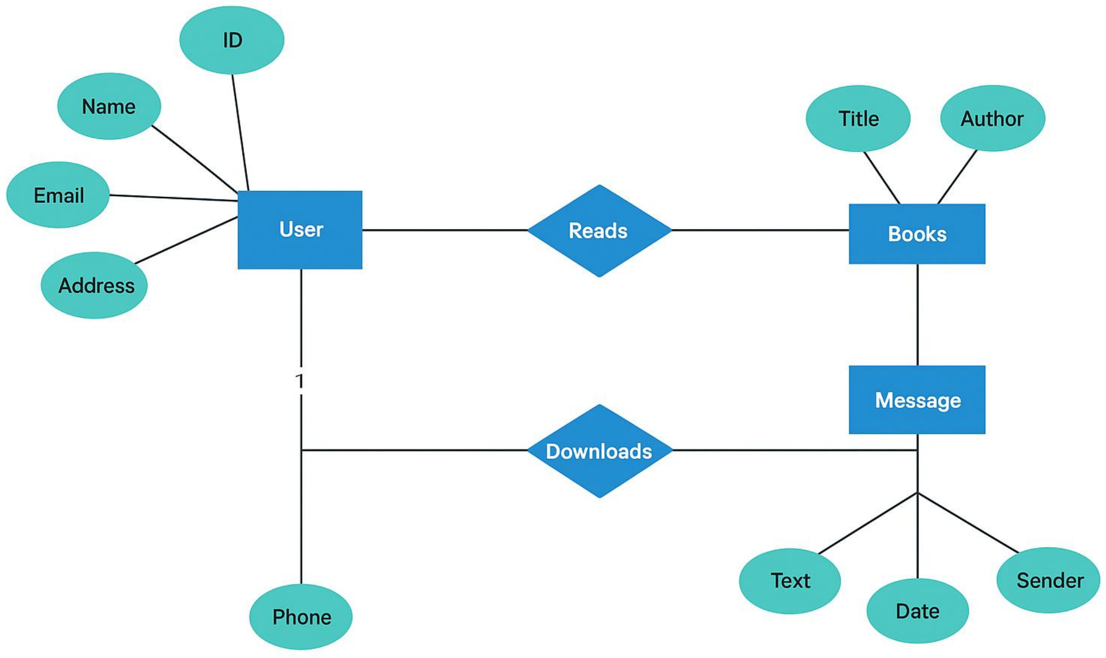
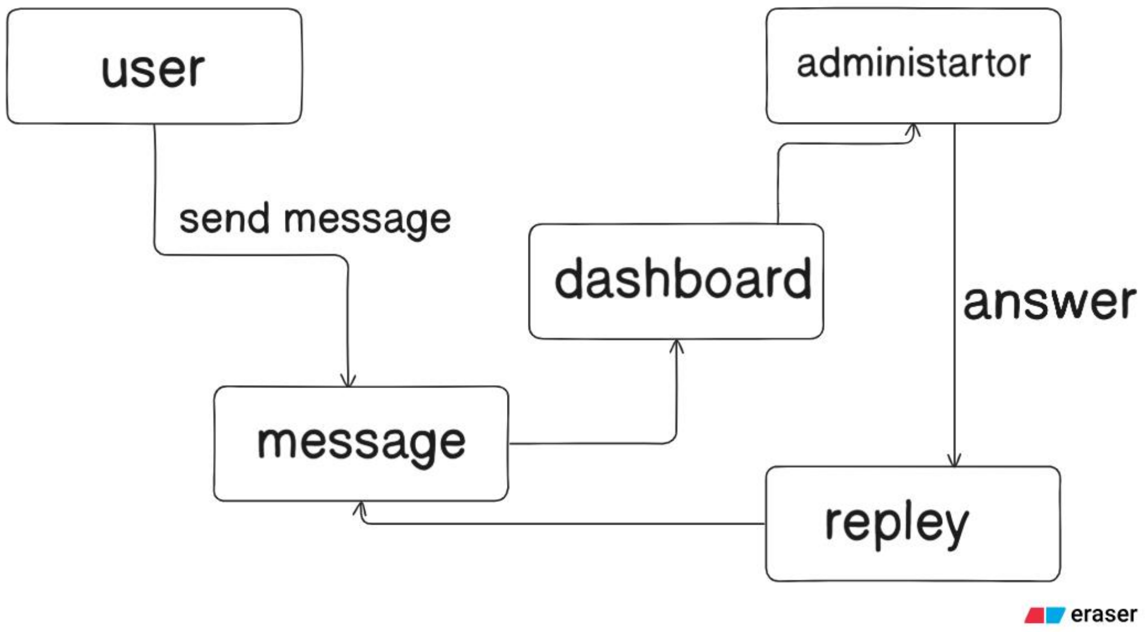


Figure 5 - Help asking Diagram

2.2.5 System Work Data Flow Diagram

DFD: System Details Diagram:



Initial Step-By-Step Description

Before this use case can be initiated, the User has already accessed the main page of the Program.

1. 2. The user has a choice of request or show a problem.
2. The user chooses to add or to show.
3. If the user is adding a new request, the system presents a list of type of problems to choose from and presents a grid filling in with the information; else the system presents a blank grid.
4. The User fills in the problem information and submits the form. 6. The system shows the information to other users for get a help.

2.3 Non-Functional Requirements (Digital Library System)

• **Performance**

When a user searches or requests a book, the system should respond within a few seconds to ensure fast and efficient performance.

• **Usability**

The system should be simple, clear, and easy to use for all users including students, librarians, and administrators.

• **Accessibility**

Users should be able to access the digital library anytime and from any device connected to the internet.

• **Scalability**

The system should allow the administrator to add more users, books, or categories without affecting its performance.

• Interoperability

The system should be able to integrate with other educational or library management systems if needed.

• Reliability

the system should operate correctly at all times, ensuring users can browse, read, and download books without interruptions.

• Maintainability

The system should be easy to update and fix whenever technical issues occur, ensuring continuous operation.

• Serviceability

Users should be able to contact the administrator for assistance, feedback, or reporting problems through the system.

• Security

The system should protect user data and books from unauthorized access, hacking, or viruses.

• Regulatory

All operations within the digital library should follow institutional policies and ensure ethical use of digital materials.

• Manageability

The administrator should have full control to manage user accounts, monitor activities, and maintain the library database.

3.0 System Design

The design of the Digital Library System is based on modular architecture to enhance maintainability and scalability. It includes three layers: presentation, application, and data layers. The database will contain tables for users, books, borrow history, and admin records. Each module interacts through well-defined APIs ensuring smooth communication between components.

The interface will feature a responsive design built using HTML5, CSS3, and JavaScript frameworks. The backend will use node js and for logic and MySQL for data management. System diagrams such as ER and DFD will describe relationships among entities such as Users, Books, and Transactions.

4.0 Implementation and Testing

Implementation will begin with developing the database schema, followed by backend and frontend integration. The Firebase or MySQL database will store all user data and digital content. User authentication and role-based permissions will be implemented to control access.

Testing will involve multiple stages including: Unit Testing, Integration Testing, System Testing, and User Acceptance Testing (UAT). Each function, such as login, search, and upload, will be tested for accuracy, security, and reliability. Feedback from real users will guide improvements before final deployment.

5.0 References

1. Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education. → Provides methodologies for software development and modular design principles used in the system architecture.
2. Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education. → Reference for requirements analysis, system design layers, and testing phases.
3. W3C. (2024). *HTML5 and CSS3 Standards*. World Wide Web Consortium (W3C). Retrieved from https://www.w3.org/ → Guidelines followed for building the responsive front-end interface.
4. Mozilla Developer Network (MDN). (2024). *JavaScript Documentation*. Retrieved from https://developer.mozilla.org/ → Used for dynamic web functionality and front-end scripting references.

5. Node.js Foundation. (2024). *Node.js Documentation*. Retrieved from https://nodejs.org/en/docs
→ Backend logic, API development, and server management reference.
6. MySQL. (2024). *MySQL Reference Manual*. Oracle Corporation. Retrieved from https://dev.mysql.com/doc/
→ Database schema and SQL implementation guidance.
7. Firebase Documentation. (2024). *Firebase Authentication and Cloud Database*. Google Developers. Retrieved from https://firebase.google.com/docs
→ Reference for alternative database and authentication setup.
8. ISO/IEC 25010:2011. (2011). *Systems and Software Quality Requirements and Evaluation (SQuaRE)* — System and Software Quality Models.
→ Framework used for defining non-functional requirements such as performance, reliability, and maintainability.
9. Nielsen, J. (2020). *Usability Engineering*. Morgan Kaufmann Publishers.
→ Used to guide the usability and interface design of the Digital Library System.
10. Bootstrap Documentation. (2024). *Bootstrap Front-End Framework*. Retrieved from https://getbootstrap.com/docs/
→ Reference for implementing responsive UI design.
11. GitHub. (2024). *Open Source Digital Library Projects*. Retrieved from https://github.com/topics/digital-library
→ Used as inspiration for the structure and open-source standards.