HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

FACULTY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

Selected Topics in Computer Science Assignment 2

Group 5 Members	Id Number
1. **Eyob Ayele**	NaScR/0797/12
2. **Sewunet Woreta**	NaScR/1850/12
3. **Biruk Getachew**	NaScR/0527/12
4. **Abdulhakim Obsina**	NaScR/0040/12
5. **Bethelhem Gutu**	NaScR/0443/12

Submission to: Dr. Degif T.

Submission date: April 13, 2023

Introduction

The Library Management System (LMS) is a software system that helps manage the operations of a library. It is designed to handle tasks such as cataloging, circulation, and inventory management. The primary goal of this system is to provide an efficient and user-friendly way to manage library resources and services. The LMS is being developed as a web application using HTML, CSS and javascript as the frontend PHP and MySQL as the backend database.

The system design document (SDD) is a comprehensive guide that outlines the design and architecture of the system being developed. During this phase, the detailed design of the system is created, including the software, hardware, and network components. The design will take into account the functional and non-functional requirements gathered during the analysis phase, as well as the use cases and class diagrams developed. The goal of this phase is to create a design that is both functional and efficient, while also being easy to maintain and modify as needed. Overall, the SDD is an essential part of the library management system development process that ensures that the system is built to the highest standards of quality and performance. It provides a clear and detailed understanding of the system's design, making it easier for us the developers, stakeholders, and users to understand and use. The system will be designed using the Model-View-Controller (MVC) architectural pattern to achieve separation of concerns and maintainability of the codebase. It includes an overview of the system, requirements specification, system architecture, subsystems and their decompositions, and architecture used.

Scope of the Project:

The Library Management System (LMS) is a software application that is designed to manage the functionalities of a library. The system will be used by the library staff, students, and teachers to manage and maintain the books and other resources of the library. The LMS will be a web-based application that can be accessed from any location.

Objectives

The main objective of the Library Management System is to provide a comprehensive and easy-to-use system that can manage the library's resources efficiently. The system will allow the library

staff to manage the books and other resources, and provide access to students and teachers to browse and request books. The objectives of the system are:

- To automate the library system and reduce the manual work of library staff.
- To provide students and teachers with easy access to the library resources.
- To manage the circulation of books and keep track of the books issued to students and teachers.
- To manage the book inventory and keep track of the books available in the library.
- To generate reports and statistics for library management.

Functional requirements

Functional requirements for this library management system include:

- Ability for users to search for books and e-books by title, author, or category.
- Ability for students and teachers to borrow out books.
- Ability for circulation staff to manage the library collection by adding new materials, editing existing records, and removing materials that are no longer needed.
- Ability for circulation staff to check in and check out materials, keeping track of the materials that are currently checked out and those that are overdue.
- Ability for admin to manage user access to the system by creating, modifying, and deleting user accounts for students, teachers, and circulation staff.
- Ability for admin to manage permissions for different users, such as giving circulation staff access to certain features and functionalities, while limiting access for students and teachers.
- Ability for students and teachers to access e-books.
- Ability for the system to provide security features to prevent unauthorized access, data loss, and data breaches and prevent users from downloading books from the e-resource section, copying, and pasting text, or taking screen shots.

Non-functional requirements

Non-functional requirements for this library management system include:

- Usability: the system is user-friendly and intuitive, so that users and staff can use it effectively and efficiently.
- Scalability: the system is scalable, meaning it can be easily expanded to accommodate the growing needs of the library.
- Compatibility: the system is compatible with different platforms and devices, allowing users to access the system from anywhere, at any time.
- Security: the system is secure and comply with legal and regulatory requirements, such as data privacy, security, and accessibility.
- Availability: the system is highly available and can handle a high number of concurrent users.
- Performance: the system is fast and responsive to ensure a seamless experience for the users.
- Maintainability: the system is easily maintainable and upgradable.

Use case diagram

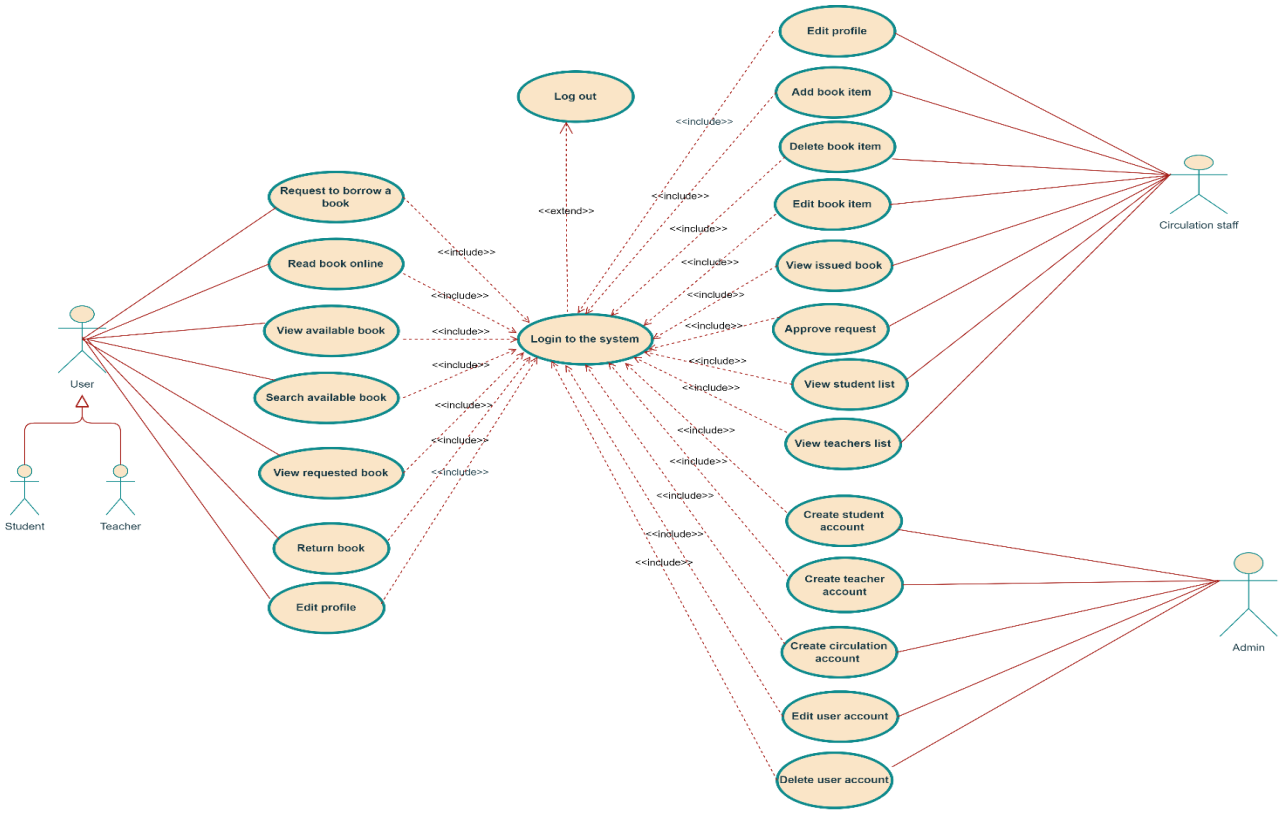

Figure 1. Use case diagram

High-level architecture of the system, including the hardware and software components, as well as the interactions between them.

The high-level architecture of the Library Management System will consist of the following components:

User Interface (UI) - This component will provide a graphical user interface for the users to interact with the system. The UI will be implemented using HTML, CSS, and JavaScript.

Application Logic (AL) - This component will contain the business logic of the system. It will be responsible for handling user requests, performing database operations, and updating the user interface. The AL will be implemented using PHP.

Database Management System (DBMS) - This component will store and manage all the data related to the system, including user information, book information, and transaction information. The DBMS will be implemented using MySQL.

The interactions between these components are as follows:

- The user interacts with the system through the UI, which sends requests to the AL.
- The AL processes the requests and interacts with the DBMS to retrieve or update data.
- The AL updates the UI with the results of the requested operation.

The system will be deployed on a web server, which will provide the necessary hardware and software resources to run the system. The hardware and software requirements for the server will be:

- A web server software called Apache
- PHP MySQL
- Adequate processing power, memory, and storage to handle the expected number of users and data volume.

The high-level architecture of the system will be designed to be scalable and maintainable, with separation of concerns between the components, modularity, and clear interfaces between the components.

Subsystems and their decompositions of the project.

The subsystems and their decompositions of a library management system are:

User Management Subsystem

- Create User Account
- Edit User Account
- Delete User Account

Authentication and Authorization Subsystem

- User Login
- User Logout
- User Authentication
- User Authorization

Book Management Subsystem

- Add Book
- Edit Book
- Delete Book
- View Book Details
- Search Book
- Book Checkout

Request Management Subsystem

- Request Book
- Approve Book Request
- Reject Book Request
- Cancel Book Request
- View Requested Books

User Profile Management Subsystem

- View Profile
- Edit Profile
- Change Password

- Reset Password

Staff Management Subsystem

- View List of Students
- View List of Teachers
- View List of Requested Books
- Approve Book Checkout

Reporting Subsystem

- Generate Reports
- Export Reports
- View Statistics

Architectural Design

We have chosen the Model-View-Controller (MVC) architectural style for the project because:

1. Separation of concerns: MVC architecture separates the application into three distinct components: the model, which represents the data and business logic; the view, which presents the data to the user; and the controller, which handles user input and updates the model and view accordingly. This separation of concerns makes it easier to maintain and modify the application over time.
2. Flexibility: The MVC architecture provides a high degree of flexibility, allowing us to easily swap out components or modify the application without affecting the rest of the system.
3. Scalability: By separating the application into distinct components, MVC architecture makes it easier to scale our system as needed. For example, if the application requires more processing power to handle increased traffic, we can scale up the server hosting the model without affecting the view or controller components.
4. Testability: Because the MVC architecture separates the application into distinct components, it is easier to write unit tests for each component. This makes it easier to test the application as a whole and ensure that each component is functioning as expected.

5. Reusability: The MVC architecture promotes code reuse by allowing us to reuse components across different parts of the application. For example, the same model component could be used in multiple views or controllers, making it easier to develop and maintain the application over time.

In the context of our library management system project, the Model-View-Controller (MVC) architectural pattern can be described as follows:

Model: The model component is responsible for handling the data and the business logic of the application. In the library management system, the model component includes:

- Book Model: It represents the data structure of the book object, including its attributes such as title, author, publisher, publication year, ISBN, and availability status.
- User Model: It represents the data structure of the user object, including attributes such as name, username, password, email, and role (admin or user).
- Loan Model: It represents the data structure of the loan object, including attributes such as the user who borrowed the book, the book that was borrowed, the date borrowed, and the due date.
- Database Connection: It establishes a connection to the database and provides methods for performing CRUD (Create, Read, Update, Delete) operations on the book, user, and loan data.

View: The view component is responsible for rendering the user interface of the application. It displays the data from the model component to the user and receives user input for processing. In the library management system, the view component includes:

- login page
- Dashboard page
- search page for books
- book details page
- Request book page
- E-books page
- Approve requests page.
- Account management page

- Admin page for managing user accounts.

Controller: The controller component acts as the mediator between the model and the view components. It receives user input from the view component, processes it, and updates the model component accordingly. The controller also communicates with the view component to update the user interface based on changes in the model component. In the library management system, the controllers would include classes such as BookController, AuthorController, UserController, BorrowerController, etc. These classes would handle user requests, such as searching for a book or borrowing a book, and communicate with the appropriate models to retrieve or update the data. They would also interact with the views to display results to the user. The controllers for the library management system include:

- BookController: Handles requests related to books, such as adding new books, editing book details, deleting books, and searching for books.
- UserController: Handles requests related to users, such as adding new users, editing user details, deleting users, and searching for users.
- BorrowController: Handles requests related to borrowing and returning books, such as issuing a book to a user, returning a book, and checking the availability of a book.
- LoginController: Handles requests related to user authentication and authorization, such as logging in and logging out of the system, and validating user credentials.
- DashboardController: Handles requests related to the main dashboard of the system, such as displaying statistical data, generating reports, and managing system settings.

In the MVC architecture, the views are completely separate from the models and controllers. The models and controllers can communicate with each other, but they should not communicate directly with the views. Instead, the views should receive their data from the controllers, and the controllers should update the models based on user input received from the views. This separation of concerns makes the application easier to develop and maintain, as changes can be made to one component without affecting the others.

Logical view of the architecture

The logical view is an important aspect of the system design, as it defines the architecture of the system and provides a basis for verifying that the system meets its functional and non-functional

requirements. The logical view is often described using diagrams and models that illustrate the system's components and their interactions.

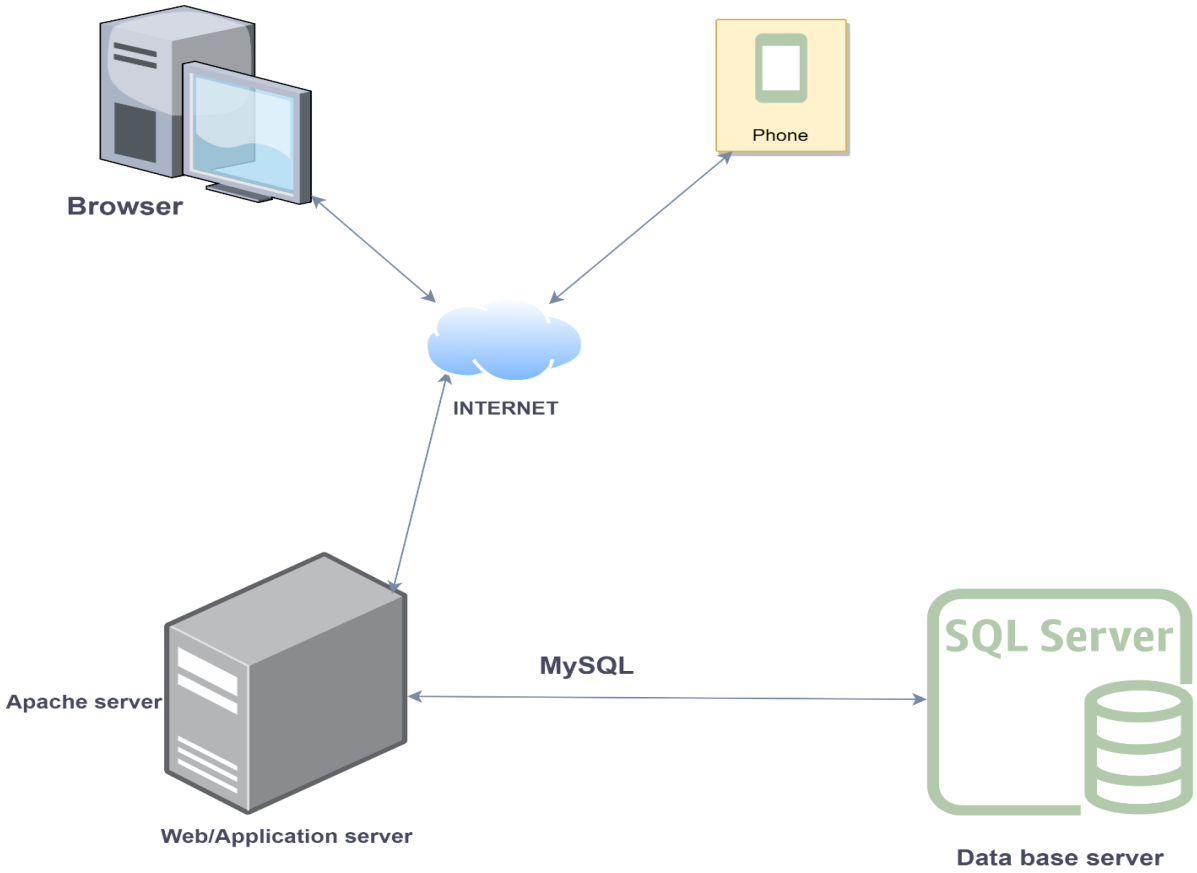

Figure 2. Logical view

Summary

The design document describes a library management system that aims to automate and improve the book management process in a university library. The document starts with an introduction, followed by a scope section that outlines the project's objectives, goals, and limitations. The high-level architecture section describes the hardware and software components used in the system and their interactions. The design document emphasizes the use of the MVC architecture pattern to improve the system's maintainability and scalability.