Library Management System - Architecture Design Document

Introduction

Purpose

This document describes the architecture of the Library Management System (LMS), focusing on how the system is structured to support its functionalities efficiently.

1.2 Scope

The system is a web-based application that enables:

- Book cataloging
- Reserving Books
- User management
- Borrowing/returning books
- Fine management

2. Architectural Overview

The LMS follows a **Layered (3-Tier) Architecture**, which ensures modularity, scalability, and security. The system is divided into three main layers:

Presentation Layer (Front-End)

Technologies: HTML, CSS, JavaScript (React/Vue/Angular), Java (Spring Boot)

Responsibilities:

- Provides an intuitive **Graphical User Interface (GUI)** for users.
- Displays book details, borrowing status, and overdue notifications.

Business Logic Layer (Back-End)

Technologies: Java (Spring Boot) or Python (Django/Flask)

Responsibilities:

- Handles **user authentication, reservation of books, book transactions, fine calculations, and book recommendations**.
- Implements **role-based access control (librarians vs. students)**.
- Processes requests from the **Front-End** and interacts with the **Database Layer**.

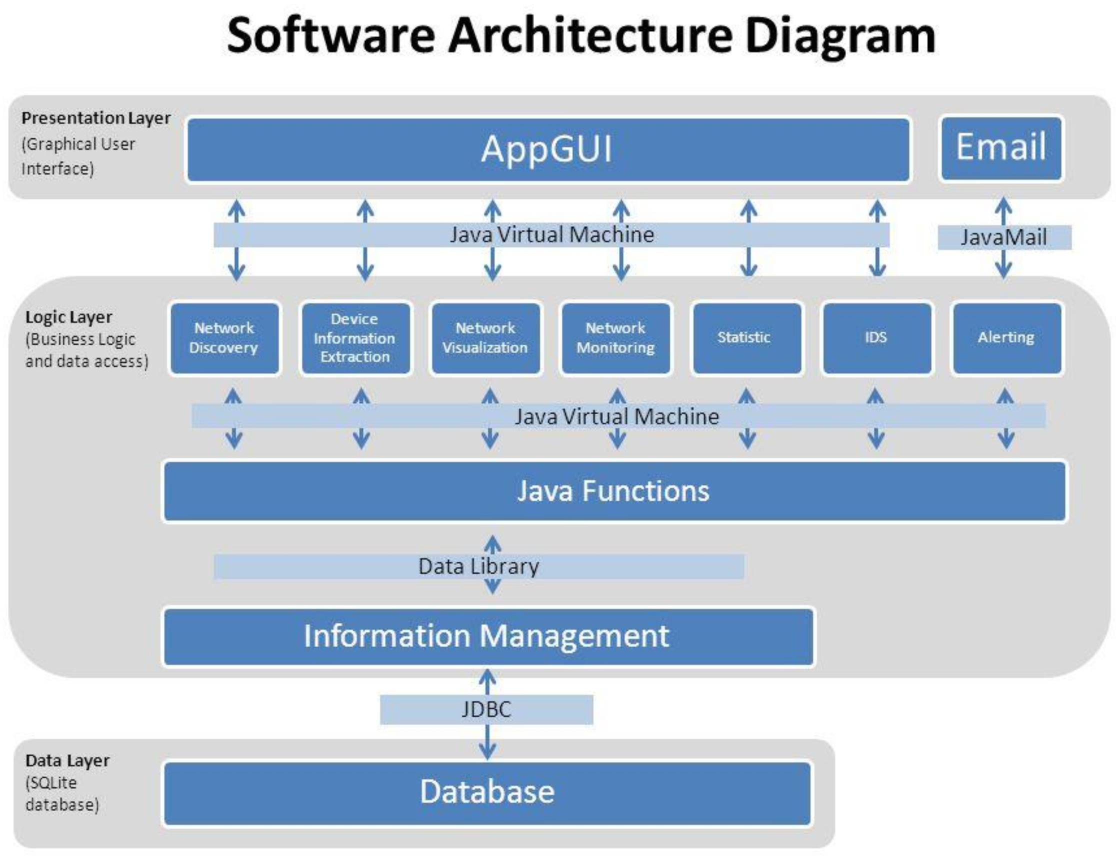
Data Layer (Database)

Technology: MySQL (RDBMS)

Responsibilities:

- Stores **book records, user information, borrowing history, and fines**.
- Ensures **data integrity, indexing, and fast retrieval**.
- Supports **data export (CSV/PDF) and backup mechanisms**.

3. System Architecture Diagram



4. Justification for Layered Architecture

Why 3-Tier Architecture?

- **Separation of Concerns** → Each layer has a dedicated role, making development and debugging easier.
- **Scalability** → Can integrate mobile apps or expand features in the future.
- **Security** → Authentication and database management are handled separately, reducing risks.
- **Maintainability** → Updates can be made in one layer without affecting the others.

5. Technologies & Tools Used

Component	Technology
Front-End (UI)	HTML, CSS, JavaScript
Back-End (API)	Java (Spring Boot) / Python (Django)
Database	MySQL
Authentication	JWT (JSON Web Tokens)
Deployment (Optional)	Docker / AWS / Heroku

6. Future Enhancements

- Implement **real-time notifications** (e.g., book due reminders) using WebSockets.
- Enhance **book recommendation system** with **Machine Learning (ML)**.

7. Conclusion

The **Library Management System** is built using a **Layered (3-Tier) Architecture**, ensuring a **modular, scalable, and secure** design. This document provides a structured approach to understanding and implementing the system efficiently.

Next Steps:

- Start implementing the **database schema**.
- Build the **frontend UI** and integrate it with the backend.