

## **Library Management System**

A Library Management System is software that provides the ability to find books, manage books, track borrowed books, managing fines and bills all in one place. It helps the librarian manage the books and books borrowed by members and automates most of the library activities. It increases efficiency and reduces the cost needed for maintaining a library and saves time and effort for both the user and the librarian.

## **System Requirements**

The key requirements that need to be offered by the library management system can be classified into functional and non-functional requirements.

### Functional Requirements

1. Allow the librarian to add and remove new members.
2. Allow the user to search for books based on title, publication date, author, etc., and find their location in the library.
3. Users can request, reserve, or renew a book.
4. Librarian can add and manage the books.
5. The system should notify the user and librarian about the overdue books.
6. The system calculates the fine for overdue books on their return.

A more detailed list of key features that need to be supported by the system is given in the use case diagram.



There are 3 actors in the use case diagram. The User, The Librarian, and the System.

**User** - The user can log in, view the catalog, search for books, checkout, reserve, renew and return a book.

**Librarian** - The librarian registers new users, adds and maintains the books, collects fines for overdue books, and issues books to users who need them.

## Non - Functional Requirements

### **Usability**

Usability is the main non-functional requirement for a library management system. The UI should be simple enough for everyone to understand and get the relevant information without any special training. Different languages can be provided based on the requirements.

### **Accuracy**

Accuracy is another important non-functional requirement for the library management system. The data stored about the books and the fines calculated should be correct, consistent, and reliable.

### **Availability**

The System should be available for the duration when the library operates and must be recovered within an hour or less if it fails. The system should respond to the requests within two seconds or less.

### **Maintainability**

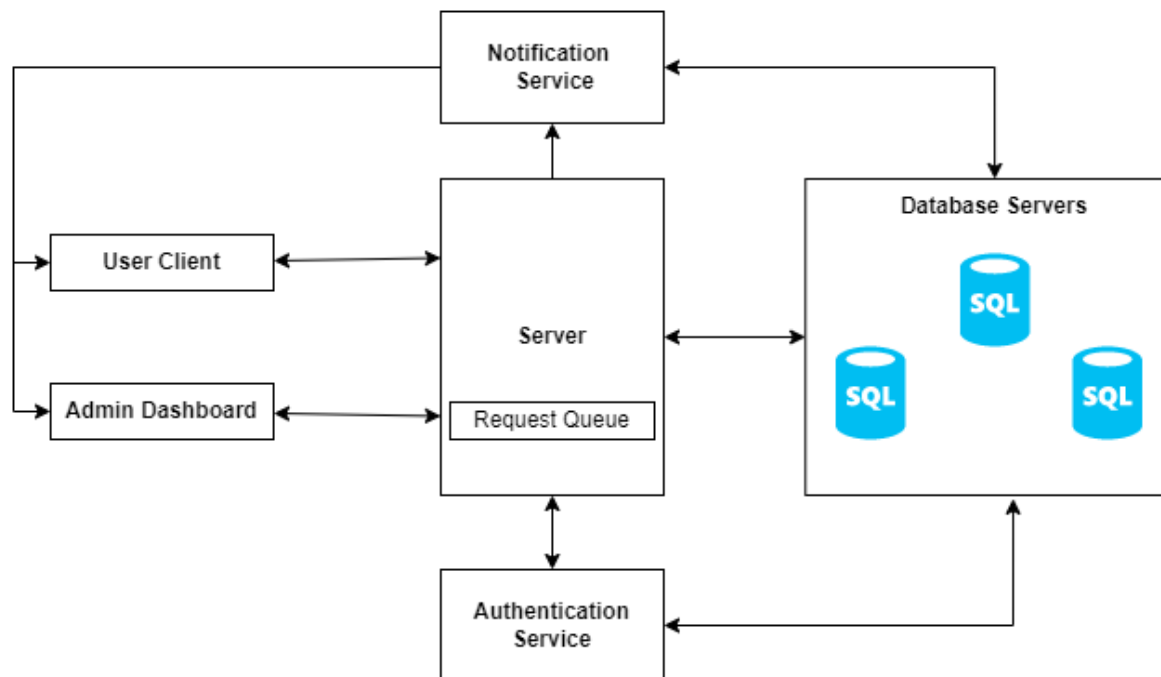
The software should be easily maintainable and adding new features and making changes to the software must be as simple as possible. In addition to this, the software must also be portable.

## Software Requirements

1. A server running Windows Server/Linux OS
2. A multi-threading capable backend language like Java
3. Front-end frameworks like Angular/React/Vue for the client
4. Relational DBMS like MySQL, PostgreSQL, etc
5. Containers and orchestration services like Kubernetes (for a large setting like a national library).

## **High Level Design**

## For a Small Setting (School/College):



iq.opengenus.org

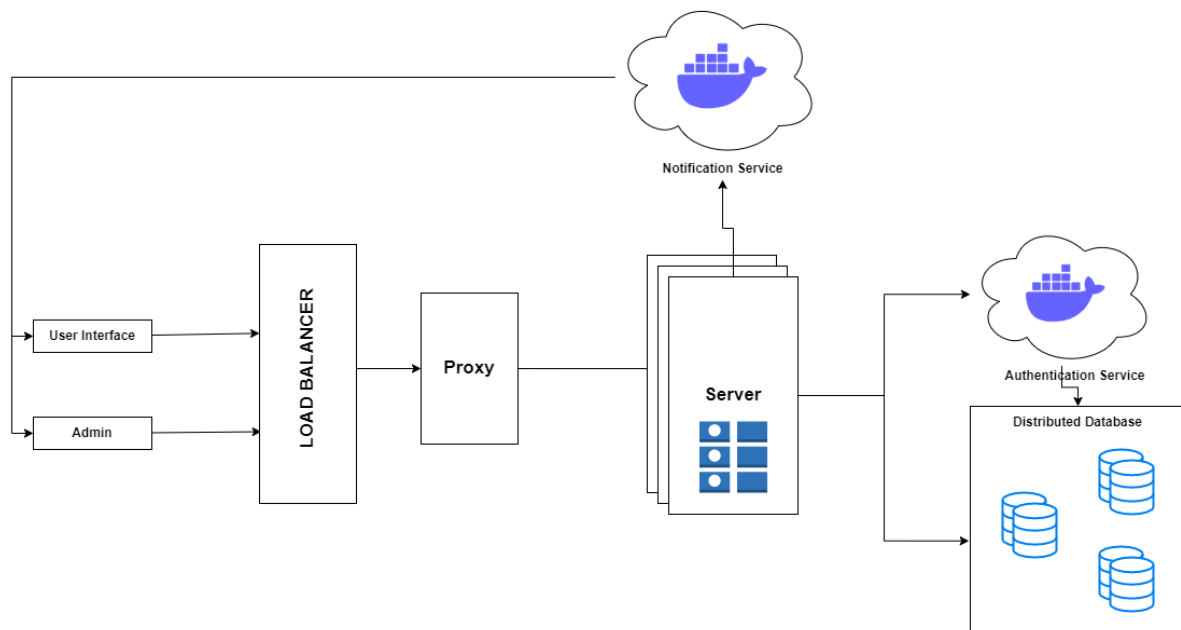
In a school or college, The users are usually students and the faculty of the institution. Hence, the traffic is usually pretty low with an estimated 5,000 concurrent users at the peak. A group of servers each with 16GB of RAM and a capable processor should be able to handle the load. Each server may need to handle several requests at a time. The requests can be queued and a reply can be sent to the user while the request is being processed.

The user can access the system through a client site or app with a registered account. The librarian can access the admin dashboard and interface using the admin account.

Handling the authentication is done by a separate service. The authentication service has direct access to the accounts on the database. The notification service checks the database for any overdue books and alerts the user and the librarian. It can also be used by the server to notify about book reservations and cancellations.

Since the data stored are mostly relational, like user accounts, details on books, etc., a relational database like MySQL can be used. The database servers can be set up in master-slave configuration for backup and availability.

### **For a Large Setting (State/National Library):**



iq.opengenus.org

These libraries are huge and are available to the general public. Anyone can visit these libraries. However, they may contain many important artifacts and rarely allow users to borrow books. The number of concurrent users can range from 100,000 for a state library to 10 Million for a national library at its peak.

These systems can also have separate user and admin interfaces for browsing books and managing users and books respectively.

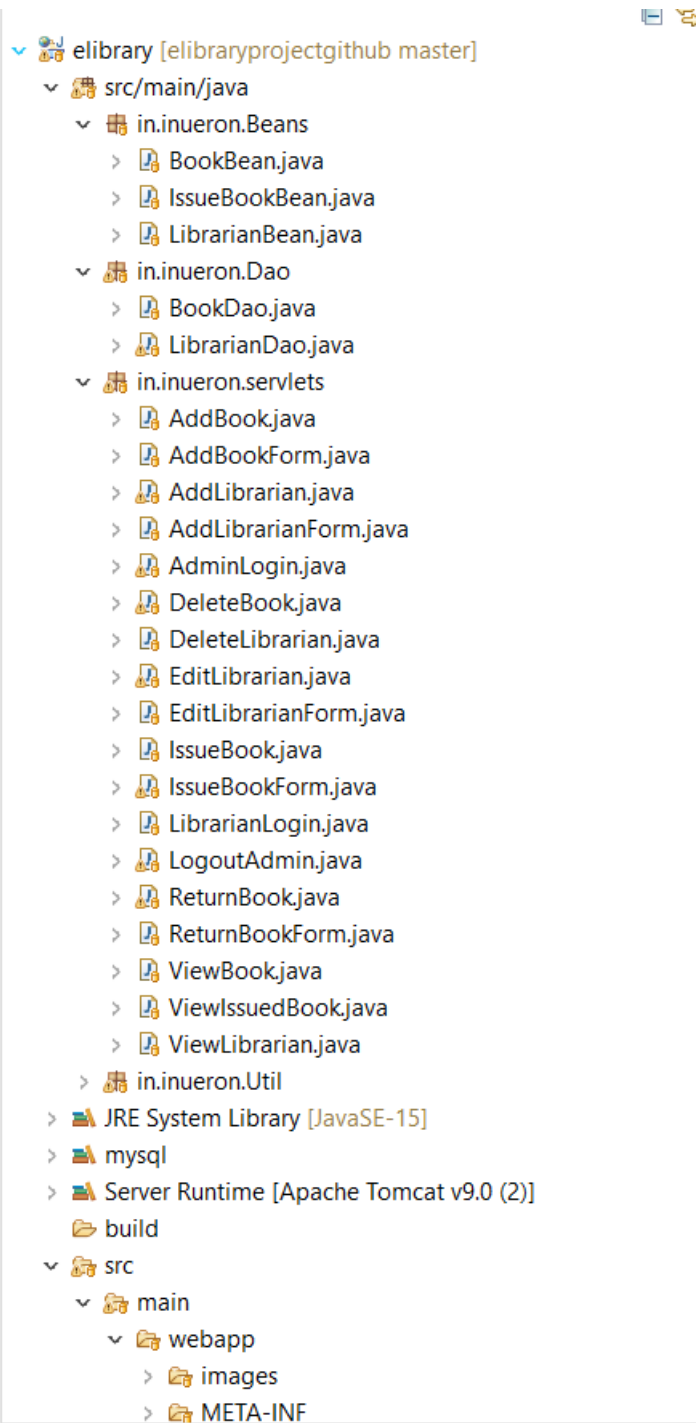
The servers may be distributed across different regions in the country with load balancers to distribute the traffic. Proxy servers may be used to cache the requests and reduce the number of disk reads. Services such as authentication and notification may be deployed as


























separate containers. This allows scaling much more easily and simplifies the addition of new services.

The database can also be distributed with the regional data such as books of languages spoken in a region being stored in that region. Many government libraries preserve physical books in digital form and allow anyone to read them online without borrowing.

## **Low-Level Design and Classes**

Now, let us take a look at a little bit lower-level design of the library management system by exploring the various classes and methods involved.



- >  in.inueron.Util
- >  JRE System Library [JavaSE-15]
- >  mysql
- >  Server Runtime [Apache Tomcat v9.0 (2)]
-  build
-  src
  -  main
    -  webapp
      - >  images
      - >  META-INF
      - >  WEB-INF
        -  addbookform.html
        -  addlibrarianform.html
        -  admincarousel.html
        -  bootstrap.min.css
        -  bootstrap.min.js
        -  footer.html
        -  index.html
        -  issuebookform.html
        -  jquery.min.js
        -  librariancarousel.html
        -  LibrarianForm.html
        -  navadmin.html
        -  navlibrarian.html
        -  returnbookform.html

