**SOFTWARE SPECIFICATION**

LMSs are software programs that assist libraries in managing their operations and collection management. We have modules for circulation, purchases, serials administration, reporting, and cataloguing. Automating library activities including cataloguing, circulation, acquisitions, membership management, and reporting is the major goal of a library management system. It strives to provide superior experiences for both library personnel and users while enhancing the accessibility, tracking, and consumption of library resources.

**System Overview**

**System Name: Library Management System (LMS)**

**Purpose**: The university implements a robust LMS, allowing students and faculty to access the library's resources seamlessly. The LMS addresses the challenges by offering the following benefits:

* **Efficient Book Management:** Books are digitally catalogued and categorized based on various criteria (title, author, subject, etc.), enabling quick and accurate searches.
* **Streamlined Transactions:** Borrowing and returning books are simplified through an online portal, reducing the time and effort required for these transactions.
* **Enhanced Accessibility:** The LMS is accessible remotely, enabling users to search for and reserve books from their homes or other locations. They can also receive notifications and reminders for due dates.

**Scope:**

**Catalog Management:**

- Cataloging and indexing of library materials (books, journals, DVDs, etc.).

- Tracking and managing item information (ISBN, author, publisher, publication year, etc.).

- Support for different types of materials (e.g., books, magazines, e-books, multimedia).

**Check-In and Check-Out:**

- Facilitating the borrowing and return of library materials.

- Managing due dates and fines for late returns.

- Reserving and renewing items.

**Search and Discovery:**

- Powerful search capabilities for users to find library materials.

- Advanced search options (e.g., by author, title, subject, keyword).

- Integration with external databases and online catalogs.

**Notifications and Alerts:**

- Automated notifications to patrons (e.g., due date reminders, hold pickup notices).

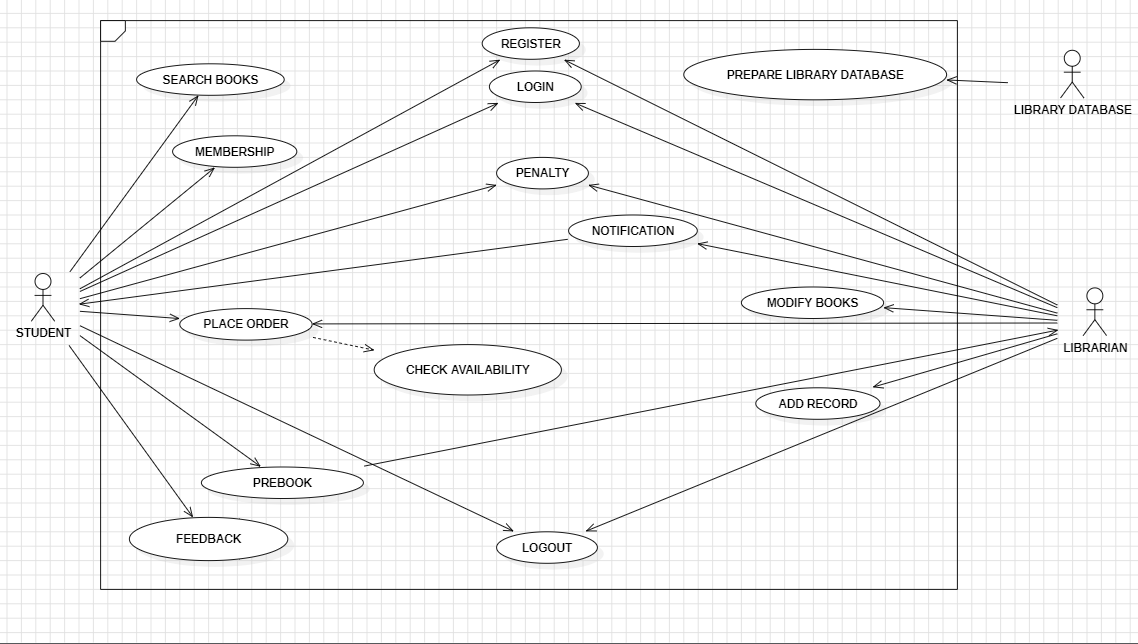
- Alerts to librarians for system-related issues.

**Stakeholders**:

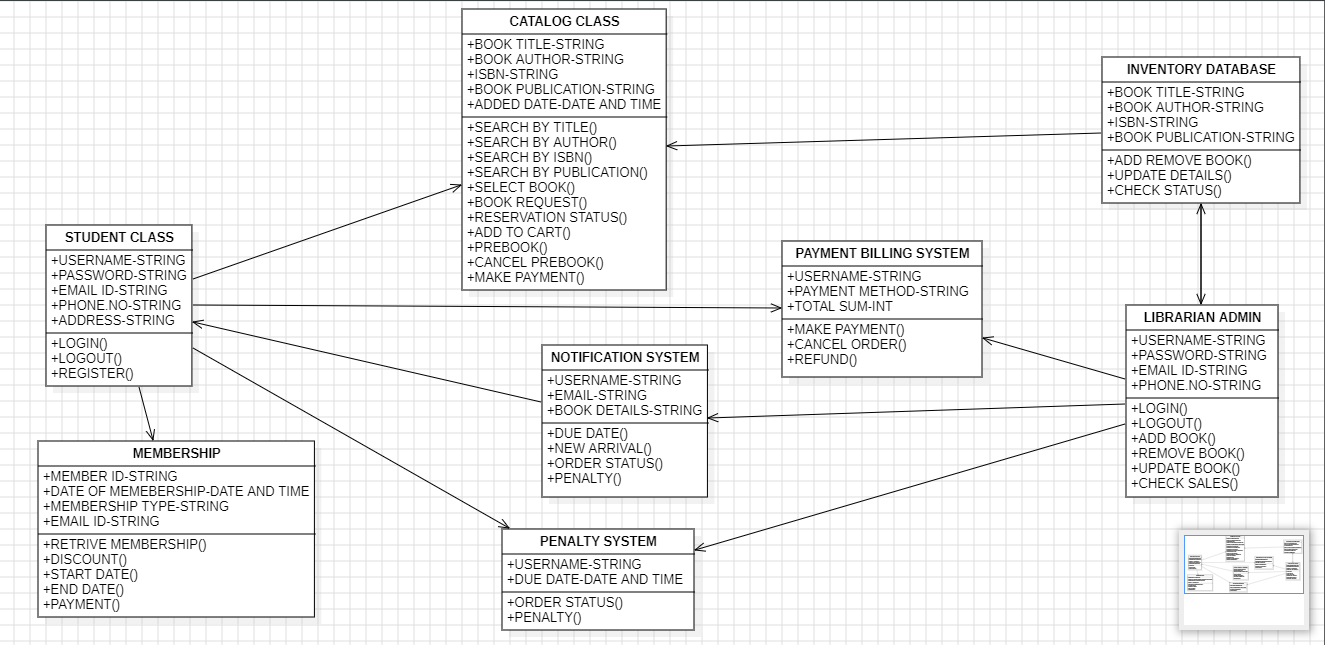
Librarians, library staff, patrons (students, researchers, or general users), administrators.

**System Architecture Diagram:**

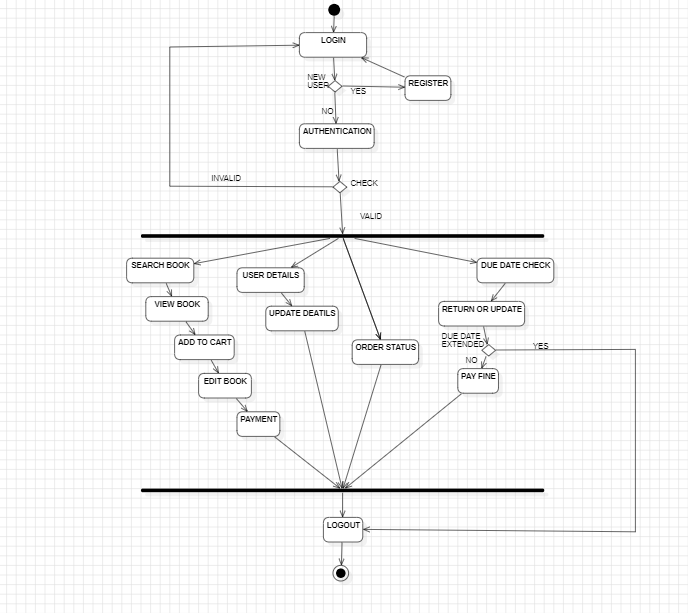
*USE-CASE DIAGRAM*



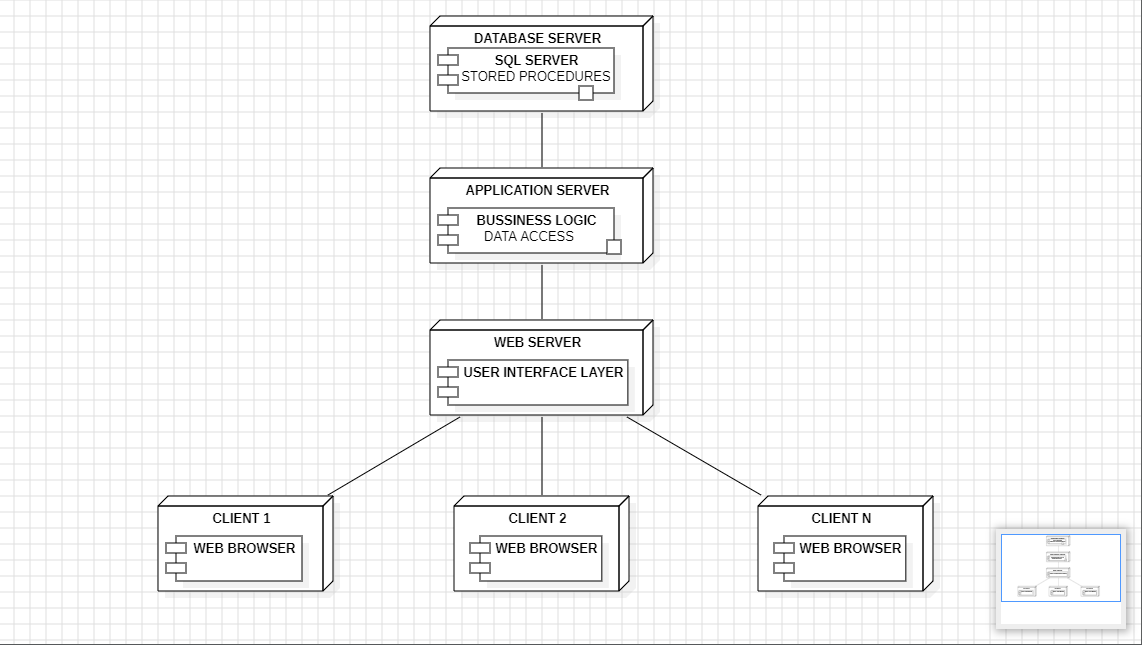
*CLASS DIAGRAM*



*ACTIVITY DIAGRAM*



*DEPLOYMENT DIAGRAM*



**Classes of Library Management System:**

* **Catalogue class –** It manages the operations related to the books present in the catalogue of the library management system including the details related to book title, book author, etc.
* **Librarian Class –** It manages all operations of Librarian.
* **Payment Billing Class –** It manages all operations of account.
* **Inventory database Class –** It manages all operations of the inventory database.
* **Membership Class –** It manages all operations related to the membership.
* **Student Class –** It manages all operations of student.
* **Penalty Class-** It stores and regulates the information related to penalty.

**Attributes of Library Management System :**

* **Catalogue class –** Book title, book author, ISBN, Book Publication, Added Date.
* **Librarian Class –** Username, Password, Email ID, Phone number
* **Payment Billing Class –** Username, Payment Method, Total Sum
* **Inventory database Class –** Book Title, Book author, ISBN, Book Publication
* **Membership Class –**  Member ID, Date of Membership, Membership Type, Email ID
* **Student Class –** Username, Password, Email ID, Phone number, Address
* **Penalty Class-** Username, Due Date

**Methods of Library Management System :**

* **Catalogue class –** SearchByTitle(), SearchByAuthor(), SearchByISBN(), SearchByPublication(), SelectBook(), BookRequest(), ReservationStatus(), AddToCart(), Prebook(), CancelPrebook(), MakePayment()
* **Librarian Class –** Login(), Logout(), AddBook(), RemoveBook(), UpdateBook(), CheckSales()
* **Payment Billing Class –** MakePayment(), CancelOrder(), Refund()
* **Inventory database Class –** Add\_Remove\_Book(), UpdateDetails(), CheckStatus()
* **Membership Class –**  RetrieveMembership(), Discount(), StartDate(), EndDate(), Payment()
* **Student Class –** Login(), Logout(), Register()
* **Penalty Class-** OrderStatus(), Penalty()

**Key Components**

**1. Backend: PHP**

**Description:** PHP (Hypertext Preprocessor) is a widely- used open-source server-side scripting language that is well-suited for web development.

**Benefits:**

- PHP is easy to learn and use, making it a popular choice for web development.

- It offers seamless integration with databases and supports a variety of databases, including MySQL.

- PHP supports a wide range of frameworks and libraries that facilitate efficient backend development.

**2. PHP Framework: Laravel or CodeIgniter**

**Description:**

**Laravel**: A PHP framework known for its elegant syntax, robust features, and developer-friendly tools. It follows the Model-View-Controller (MVC) architecture.

**CodeIgniter:** Another PHP framework that is lightweight and highly efficient, suitable for developing dynamic web applications.

**Benefits:**

- Frameworks like Laravel and CodeIgniter provide pre-built modules and tools, reducing development time and effort.

- They offer features like routing, database abstraction, authentication, and more, promoting organized and scalable code.

**3. Database Management System: MySQL**

**Description:** MySQL is a popular open-source relational database management system known for its reliability, speed, and ease of use.

**Benefits:**

- MySQL supports ACID (Atomicity, Consistency, Isolation, Durability) compliance, ensuring data integrity and reliability.

- It provides excellent performance for read-heavy applications and is widely adopted in the industry.

**4. Frontend: HTML, CSS, JavaScript**

**Description:**

**HTML (Hyper Text Markup Language)**: Used for structuring the content and elements on web pages.

**CSS (Cascading Style Sheets)**: Used for styling the appearance and layout of web pages.

**JavaScript**: A versatile scripting language used for creating interactive and dynamic web content.

**Benefits:**

- HTML provides the basic structure for web pages, CSS enhances their visual appeal, and JavaScript adds interactivity, improving the user experience.

**5. Frontend Framework or Library: React or Angular (optional)**

**Description:**

**React**: A popular JavaScript library for building interactive user interfaces, known for its component-based architecture and virtual DOM.

**Angular:** A robust JavaScript framework that facilitates building dynamic and scalable web applications.

**Benefits:**

- React and Angular help in creating efficient and engaging frontend experiences, enabling developers to build complex UIs with ease.

**6. Version Control: Git**

**Description:** Git is a distributed version control system that allows tracking changes in code during software development.

**Benefits:**

- Git enables collaboration among developers, version tracking, and easy integration of changes, promoting efficient and organized development.

7. **Authentication and Authorization: OAuth or JWT**

**Description:**

**OAuth (e.g., OAuth2):** An industry-standard protocol for secure and delegated access, commonly used for user authentication.

**JWT** (JSON Web Tokens): A compact and self-contained mechanism for securely transmitting information between parties.

**Benefits:**

- These protocols provide secure and standardized methods for user authentication and authorization in web applications.

**8. Testing: PHP Unit (for PHP)**

**Description:** PHP Unit is a popular testing framework for PHP, facilitating unit testing of PHP code.

**Benefits:**

- PHP Unit helps in writing and executing automated tests to verify the correctness and reliability of PHP code, ensuring better code quality.

**9. Deployment and Hosting: Apache or Nginx**

**Description:**

**Apache:** A widely- used open-source web server known for its reliability, flexibility, and extensive community support.

**Nginx:** A high-performance, open-source web server and reverse proxy server known for its speed and efficiency.

**Benefits:**

- Apache and Nginx are crucial for hosting PHP applications and ensuring their availability and performance.

**10. Task Automation and Package Management: Composer (for PHP)**

**Description:**

**Composer:** A dependency manager for PHP that simplifies the process of managing libraries and packages in PHP projects.

**Benefits:**

- Composer streamlines the management of project dependencies and automates tasks, making PHP development more efficient and organized.

**Data Flow:**

- Patrons search the catalog, request books, and check them out.

- Librarians catalog new books and manage patron accounts.

- Transaction data is recorded in the database for reporting and analytics.

**System Architecture Patterns:**

- This LMS can use a three-tier architecture, separating the presentation layer, application logic layer, and data storage layer.

**Deployment Architecture:**

- Deploy the LMS on a cloud server or on-premises server infrastructure.

- Consider redundancy and scalability options for high availability.

**Security and Authentication:**

- Implement secure authentication and authorization mechanisms to protect patron and library data.

- Secure data transmission and storage.

**Performance and Scalability:**

- Optimize database queries for quick catalog searches.

- Implement caching for frequently accessed data.

- Scale horizontally to handle increased user loads.

**Monitoring and Logging:**

- Implement monitoring and logging tools to track system health, user activities, and errors.

- Use centralized logging for troubleshooting and auditing.

**Dependencies:**

- External book catalog APIs or databases for book information.

- Payment gateway for fine collection (if applicable).

- Integration with the library's existing infrastructure (e.g., RFID systems, security systems).

**Risks and Mitigations:**

- Data security risks: Encrypt data, conduct security audits, and regularly update security measures.

- System downtime: Implement backup and disaster recovery plans.

- User training: Provide user training to ensure efficient system use.

**Maintenance and Support:**

- Regularly update and maintain the system to address bugs and security vulnerabilities.

- Provide user support and training as needed.