Library Management System (LMS) Documentation

Project Overview

The Library Management System (LMS) is a software solution designed to streamline library operations, including book cataloging, user management, loan tracking, and data storage for authors, publishers, and categories. Built using Java, Maven, and MySQL, this system provides a robust backend for managing libraries efficiently.

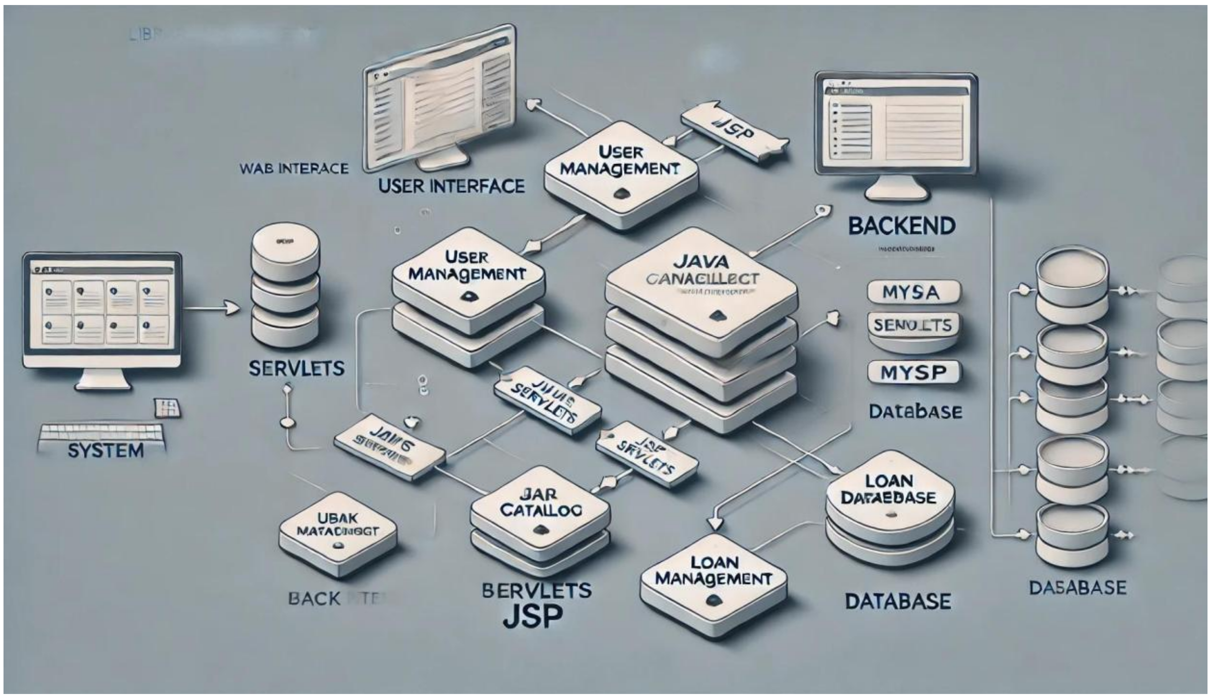
Team Members

- Nandini Singh
- Piyush Agarwal
- Aryan Sinha
- Akash Kumar Singh

Key Features

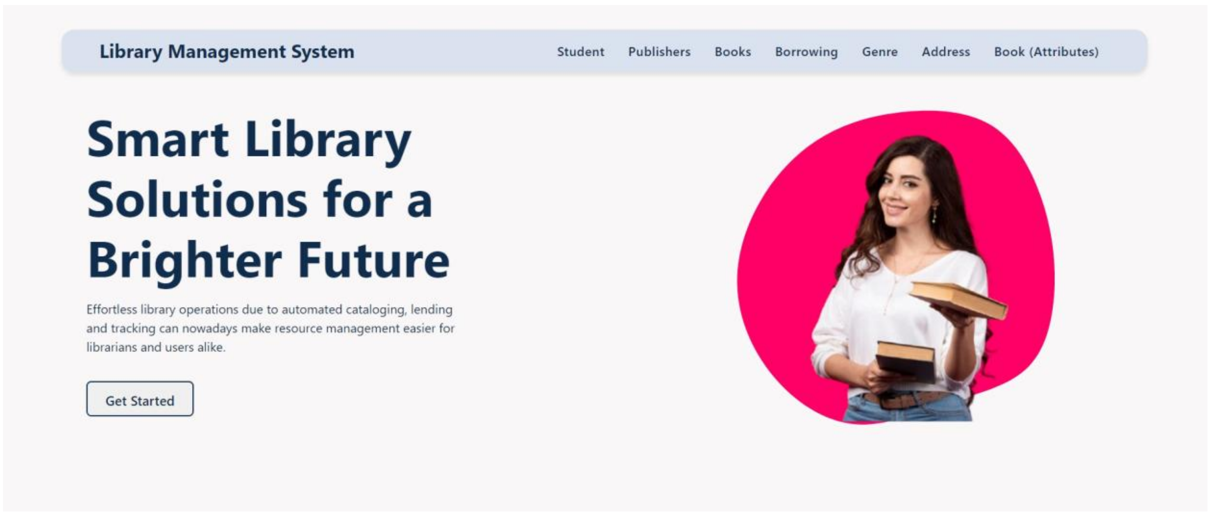
1. **User and Book Management**:
    - Manage users, authors, and books with comprehensive details such as contact information, ISBN, categories, and library locations.
2. **Efficient Loan Tracking**:
    - Real-time borrowing transaction management, including due dates, returns, and overdue handling.
3. **Relational Database Structure**:
    - Establish structured relationships between authors, publishers, categories, and books to create an organized library database.
4. **Inventory Management**:
    - Maintain book availability, condition, and stock levels.
5. **Advanced Search and Filter**:
    - Enable users to search and filter books by various attributes, such as title, author, or category.

Technologies Architectural Diagram



Technologies Used

- **Programming Language**: Java (JDK for development tools and libraries)
- **Database**: MySQL
- **Backend**:
  - Java Servlets for handling HTTP requests and responses.
  - JDBC for SQL query execution and database connections.
  - Maven for build automation and dependency management.
- **Frontend**:
  - JavaServer Pages (JSP) for dynamic content.
  - HTML & CSS for user interface design.



Smart Library Solutions for a Brighter Future

Effortless library operations due to automated cataloging, lending and tracking can nowadays make resource management easier for librarians and users alike.

Database Design

The system uses a relational database structure to manage its data effectively. Key tables include:

1. **Book Table**:
   - **Primary Key**: `book_id`
   - Fields: `title`, `isbn`, `edition`, `category_id`, `publisher_id`
2. **Persons Table**:
   - **Primary Key**: `user_id`
   - Fields: `first_name`, `last_name`, `address_id`
3. **Loan Table**:
   - **Primary Key**: `loan_id`
   - Fields: `book_id`, `reader_id`, `loan_date`

UML Diagram

The UML diagram illustrates the relationships between key entities such as books, users, loans, categories, and publishers, ensuring a clear representation of the system's architecture.

Setup Instructions

Clone Repository

```
git clone https://github.com/Nandinisingh005/Nandinisingh005-library-management-system.git
```

1. `cd library-management-system`

Database Setup

```
CREATE DATABASE lms;

USE lms;
```

2. Execute the SQL script from `SQL/lms.sql`
   ```
   mysql -u yourUsername -p library_system < lms.sql
   ```
3. **Configure Database Connection**

   Locate `resources/application.properties`

   Update with your database credentials:

```
db.url=jdbc:mysql://localhost:3306/lms

db.user=root

db.password=YOUR_PASSWORD
```

4. **Build Project**
   ```
   mvn clean install
   ```

5. **Run Application**
   ```
   mvn exec:java
   ```

API Documentation

Book Management APIs

- GET `/api/books`: Retrieve all books
- POST `/api/books`: Add new book
- PUT `/api/books/{id}`: Update book details
- DELETE `/api/books/{id}`: Remove book

User Management APIs

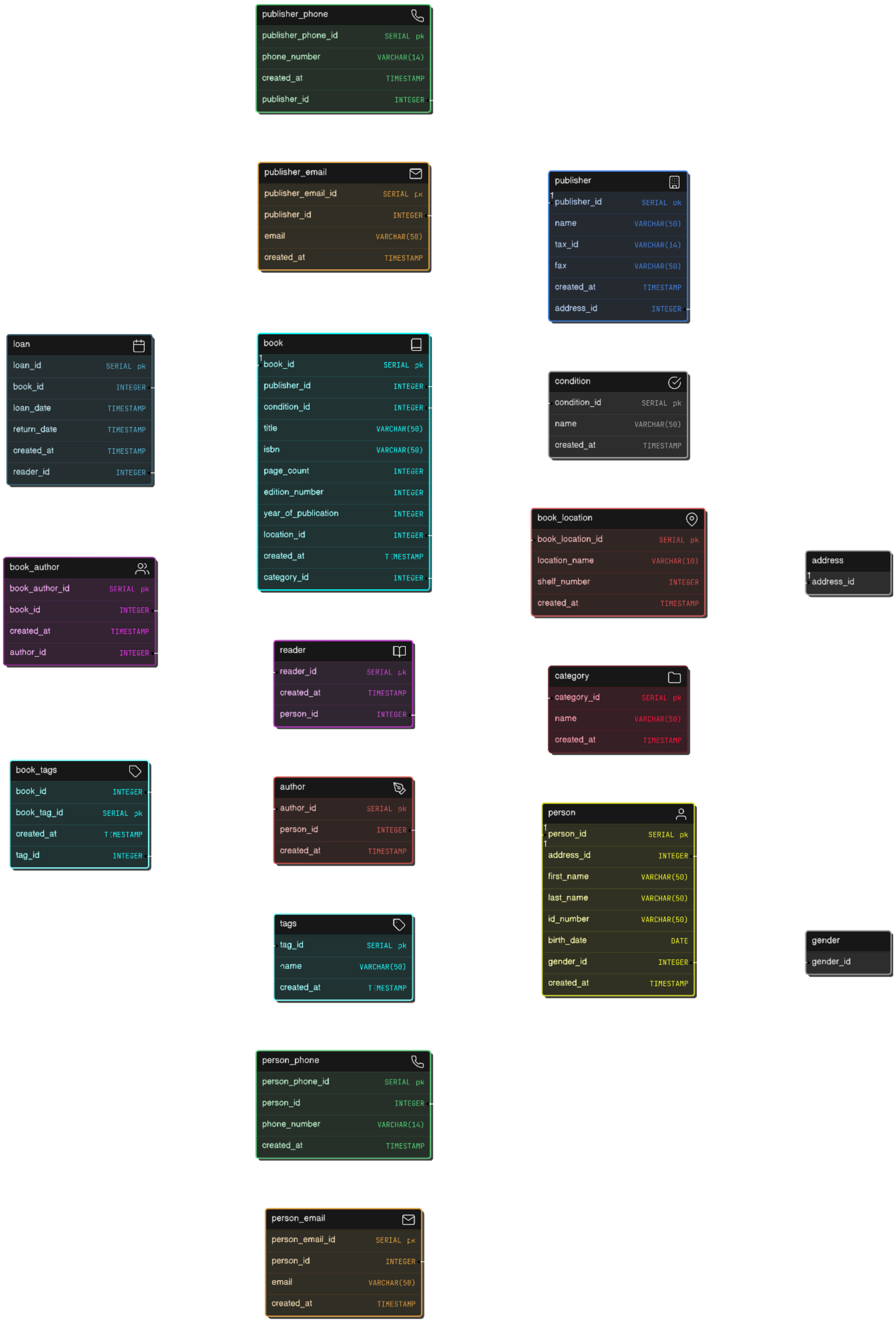
- GET `/api/users`: Retrieve all users
- POST `/api/users`: Add new user
- PUT `/api/users/{id}`: Update user details
- DELETE `/api/users/{id}`: Remove user

Loan Management APIs

- POST `/api/loans`: Create new loan
- PUT `/api/loans/{id}`: Update loan status
- GET `/api/loans/user/{userId}`: Get user's loans
- GET `/api/loans/overdue`: Get overdue loans

Additional Documentation

For further details, refer to the project UML diagrams and database schema provided in the repository.



publisher_phone	
publisher_phone_id	SERIAL pk
phone_number	VARCHAR(14)
created_at	TIMESTAMP
publisher_id	INTEGER

publisher_email	
publisher_email_id	SERIAL pk
publisher_id	INTEGER
email	VARCHAR(50)
created_at	TIMESTAMP

publisher	
publisher_id	SERIAL pk
name	VARCHAR(50)
tax_id	VARCHAR(14)
fax	VARCHAR(50)
created_at	TIMESTAMP
address_id	INTEGER

loan	
loan_id	SERIAL pk
book_id	INTEGER
loan_date	TIMESTAMP
return_date	TIMESTAMP
created_at	TIMESTAMP
reader_id	INTEGER

book	
book_id	SERIAL pk
publisher_id	INTEGER
condition_id	INTEGER
title	VARCHAR(50)
isbn	VARCHAR(50)
page_count	INTEGER
edition_number	INTEGER
year_of_publication	INTEGER
location_id	INTEGER
created_at	TIMESTAMP
category_id	INTEGER

condition	
condition_id	SERIAL pk
name	VARCHAR(50)
created_at	TIMESTAMP

book_location	
book_location_id	SERIAL pk
location_name	VARCHAR(10)
shelf_number	INTEGER
created_at	TIMESTAMP

address
address_id

book_author	
book_author_id	SERIAL pk
book_id	INTEGER
created_at	TIMESTAMP
author_id	INTEGER

reader	
reader_id	SERIAL pk
created_at	TIMESTAMP
person_id	INTEGER

category	
category_id	SERIAL pk
name	VARCHAR(50)
created_at	TIMESTAMP

book_tags	
book_id	INTEGER
book_tag_id	SERIAL pk
created_at	TIMESTAMP
tag_id	INTEGER

author	
author_id	SERIAL pk
person_id	INTEGER
created_at	TIMESTAMP

person	
person_id	SERIAL pk
address_id	INTEGER
first_name	VARCHAR(50)
last_name	VARCHAR(50)
id_number	VARCHAR(50)
birth_date	DATE
gender_id	INTEGER
created_at	TIMESTAMP

gender
gender_id

tags	
tag_id	SERIAL pk
name	VARCHAR(50)
created_at	TIMESTAMP

person_phone	
person_phone_id	SERIAL pk
person_id	INTEGER
phone_number	VARCHAR(14)
created_at	TIMESTAMP

person_email	
person_email_id	SERIAL pk
person_id	INTEGER
email	VARCHAR(50)
created_at	TIMESTAMP

Conclusion

This Library Management System offers a comprehensive solution for modern libraries, integrating advanced database relations, efficient loan management, and user-friendly features to enhance overall library operations.