Library Management System Project

Overview

The Library Management System (LMS) is a web-based application designed to automate and streamline the operations of a library. This project will utilize **Java Servlets** and **JavaServer Pages (JSP)** to create a dynamic, user-friendly system that manages books, users, and transactions such as borrowing and returning books. The system aims to provide an efficient way for librarians to manage library resources and for users (students, faculty, etc.) to interact with the library catalog.

This project is intended as an educational exercise to help students understand server-side programming, database integration, and web development using Servlets and JSP within a Java EE environment.

Objectives

- Design and implement a web application using **Servlets** for backend logic and **JSP** for the frontend interface.
- Integrate a relational database (e.g., MySQL) to store and manage library data.
- Provide role-based access: Librarian (admin) and User (borrower).
- Demonstrate CRUD (Create, Read, Update, Delete) operations for managing books and users.
- Handle session management for user authentication and authorization.

Functional Requirements

1. User Roles

- **Librarian (Admin):**
  - Manage book inventory (add, update, delete books).
  - View list of all registered users.
  - Approve or reject book borrowing/returning requests.
  - View transaction history (e.g., books borrowed, returned, overdue).
- **User (Borrower):**
  - Register and log in to the system.
  - Browse and search the library catalog.
  - Request to borrow a book.
  - Return a borrowed book.
  - View their borrowing history.

2. Core Features

- **User Authentication:**
  - Login and logout functionality using Servlets with session management.
  - Registration form for new users (stored in the database).
- **Book Management:**
  - Add a new book (title, author, ISBN, publication year, quantity).
  - Update book details.
  - Delete a book from the catalog.
  - Display all books with search/filter options (e.g., by title or author).
- **Borrowing and Returning:**
  - Users can request to borrow available books.
  - Librarian approves/rejects borrowing requests.
  - Track due dates and mark books as returned.
- **Dashboard:**
  - Librarian dashboard: Summary of total books, borrowed books, and overdue books.

- User dashboard: List of borrowed books and their due dates.

Technical Requirements

1. Technologies

- **Backend:** Java Servlets (for handling requests, business logic, and database interaction).
- **Frontend:** JSP (for dynamic HTML generation and user interface).
- **Database:** MySQL
- **Server:** Apache Tomcat (as the Servlet container).
- **IDE:** Eclipse IDE (or any preferred Java EE IDE like IntelliJ IDEA).
- **Java Version:** JDK 8 or higher.
- **Build Tool:** Maven (optional, for dependency management).

2. Development Guidelines

- Use **MVC (Model-View-Controller)** architecture:
    - **Model:** Java classes representing entities (e.g., Book, User, Transaction).
    - **View:** JSP pages for rendering the UI.
    - **Controller:** Servlets to handle requests and coordinate between Model and View.
- Implement **JDBC** (Java Database Connectivity) for database operations.
- Use **CSS** for basic styling of JSP pages (Bootstrap optional).
- Ensure proper **session management** using HttpSession for user login/logout.
- Handle exceptions and provide meaningful error messages to users.

Database Schema (Suggested)

- **Books Table:**
    - book_id (Primary Key, Auto-increment)
    - title (VARCHAR)
    - author (VARCHAR)
    - isbn (VARCHAR, Unique)
    - publication_year (INT)
    - quantity (INT)
- **Users Table:**
    - user_id (Primary Key, Auto-increment)
    - username (VARCHAR, Unique)
    - password (VARCHAR, hashed)
    - role (ENUM: 'librarian', 'user')
    - email (VARCHAR)
- **Transactions Table:**
    - transaction_id (Primary Key, Auto-increment)
    - user_id (Foreign Key referencing Users)
    - book_id (Foreign Key referencing Books)
    - borrow_date (DATE)
    - due_date (DATE)
    - return_date (DATE, nullable)
    - status (ENUM: 'pending', 'approved', 'returned', 'overdue')

Project Deliverables

1. **Source Code:**
    - Well-commented Java Servlets and JSP files.
    - SQL script for database creation and sample data insertion.

2. **Documentation:**
    - Brief report (2-3 pages) explaining:
        - System architecture (MVC).
        - Database schema with ER diagram.
        - Key features implemented.
        - Challenges faced and solutions.
3. **Demonstration:**
    - Deploy the application on Apache Tomcat.
    - Showcase all features (e.g., login, book management, borrowing process).

Learning Outcomes

- Understand how to build a dynamic web application using Servlets and JSP.
- Gain experience with database integration and SQL queries.
- Learn session management and role-based access control.
- Apply MVC architecture in a real-world scenario.
- Enhance problem-solving and debugging skills in a Java EE context.