📘 SDLC for Library Management System

✅ 1. Requirement Analysis

In this phase, the functional and non-functional requirements of the LMS are gathered.

Stakeholders:

- Library Administrator
- Librarians
- Students
- Developers

Collected Requirements:

- Book inventory management
- User authentication and roles (Admin, Librarian, Student)
- Book issue/return with due dates
- Fine calculation for overdue books
- Report generation

✅ 2. Feasibility Study

The team analyzes whether the project is **technically, financially, and operationally feasible**.

Key Outcomes:

- Chosen technology stack: PHP/Python + MySQL
- Frontend: HTML/CSS/Bootstrap
- Moderate budget and time
- Team availability confirmed

✅ 3. System Design

Design the architecture, interface, database schema, and system components.

Deliverables:

- UML diagrams: Use Case, Class, Activity, Sequence
- ER Diagram for database

- Wireframes or mockups for UI
- Defined system modules (Admin panel, Librarian panel, Student portal)

✅ 4. Implementation (Coding)

Developers start writing the actual code for each module based on design documents.

Modules:

- Admin functionalities (manage users, books, reports)
- Librarian interface (issue/return, register student)
- Student interface (search, view issued books)
- Database connection and operations (CRUD)

Technologies:

- Backend: PHP, Flask/Django
- Frontend: HTML, CSS, JS, Bootstrap
- Database: MySQL/SQLite

✅ 5. Testing

Before deployment, the system is thoroughly tested for bugs, performance, and security.

Types of Testing:

- Unit Testing (test individual functions)
- Integration Testing (verify module interactions)
- System Testing (test the entire application)
- User Acceptance Testing (UAT by librarians/admins)

✅ 6. Deployment

Once approved, the LMS is deployed on a local server or cloud platform (e.g., XAMPP, Heroku, or custom hosting).

Deployment Tasks:

- Configure web server and database
- Upload source code
- Conduct final testing in live environment

✅ 7. Maintenance

Ongoing support is provided to fix bugs, update features, and adapt to user feedback.

Examples:

- Adding search filters
- Enhancing mobile responsiveness
- Backing up data periodically