Library Management System – Complete Guide

1. Introduction

A **Library Management System (LMS)** is a software application that helps manage books, users, and transactions in a library. It allows library staff and members to:

- Add, update, or remove books
- Track issued and returned books
- Manage user registrations
- Search and filter books easily

This guide covers **requirements, steps, and full code** using **HTML, CSS, JavaScript, and PHP with MySQL**.

---

2. Requirements

Software: - XAMPP / WAMP / MAMP (Local server with PHP & MySQL) - Code Editor (VS Code, Sublime Text, etc.) - Web Browser (Chrome, Firefox)

Skills Required: - Basic HTML, CSS, JavaScript - PHP & MySQL - Understanding CRUD operations (Create, Read, Update, Delete)

---

3. Database Design

Database Name: `library_db`

Tables:

1. **users** | Field | Type | Description | |---------------|--------------|---------------------------| | user_id | INT PK AI | Primary Key | | username | VARCHAR(50) | User login name | | email | VARCHAR(100) | User email | | password | VARCHAR(255) | Hashed password | | role | ENUM('admin','member') | Role of user |

2. **books** | Field | Type | Description | |---------------|--------------|---------------------------| | book_id | INT PK AI | Primary Key | | title | VARCHAR(100) | Book title | | author | VARCHAR(100) | Book author | | genre | VARCHAR(50) | Genre | | available | INT | Number of copies |

3. **transactions** | Field | Type | Description | |---------------|--------------|---------------------------| | trans_id | INT PK AI | Primary Key | | user_id | INT FK | Reference to users | | book_id | INT FK | Reference to books | | issue_date | DATE | Date of issue | | return_date | DATE | Date of return | | status | ENUM('issued','returned') | Status of book |

---

4. Steps to Create the System

Step 1: Setup Database

```sql
CREATE DATABASE library_db;

USE library_db;

CREATE TABLE users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    role ENUM('admin','member') DEFAULT 'member'
);

CREATE TABLE books (
    book_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    author VARCHAR(100) NOT NULL,
    genre VARCHAR(50),
    available INT DEFAULT 1
);

CREATE TABLE transactions (
    trans_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    book_id INT NOT NULL,
    issue_date DATE NOT NULL,
    return_date DATE,
    status ENUM('issued','returned') DEFAULT 'issued',
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (book_id) REFERENCES books(book_id)
);
```

Step 2: Create Frontend Pages

HTML Files: - `index.html` – Home page - `login.php` – Login form - `register.php` – User registration - `books.php` – View all books - `issue_book.php` – Issue books - `return_book.php` – Return books - `admin_panel.php` – Admin management

Example: `login.php`

```php
<?php
session_start();
include('config.php');

if(isset($_POST['login'])){
    $email = $_POST['email'];
    $password = $_POST['password'];

    $query = "SELECT * FROM users WHERE email='$email'";
    $result = mysqli_query($conn, $query);
    $user = mysqli_fetch_assoc($result);

    if(password_verify($password, $user['password'])){
        $_SESSION['user_id'] = $user['user_id'];
        $_SESSION['role'] = $user['role'];
        header('Location: books.php');
    } else {
        echo "Invalid email or password.";
    }
}
?>
<form method="POST">
    Email: <input type="email" name="email" required><br>
    Password: <input type="password" name="password" required><br>
    <button name="login">Login</button>
</form>
```

Step 3: Create Backend Logic

PHP & MySQL Connection (`config.php`)

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "library_db";

$conn = mysqli_connect($servername, $username, $password, $dbname);

if(!$conn){
    die("Connection failed: " . mysqli_connect_error());
}
?>
```

Book List (`books.php`)

```php
<?php
include('config.php');

$sql = "SELECT * FROM books";
$result = mysqli_query($conn, $sql);

while($row = mysqli_fetch_assoc($result)){
    echo "Title: ".$row['title']." | Author: ".$row['author']." | Available: ".
$row['available']."<br>";
}
?>
```

Step 4: Additional Features

- **Search books:** Add a search form and query with `LIKE` in SQL
- **Role-based access:** Admin can add/remove books; members can only view and issue books
- **Issue/Return:** Update `transactions` table and decrease/increase `available` field in `books` table

---

Step 5: Testing

- Test all pages on XAMPP localhost
- Check user registration, login, book issuance, and return
- Ensure role-based restrictions are working

Step 6: Security Tips

- Always use `password_hash()` for storing passwords
- Validate inputs to prevent SQL injection
- Use sessions to manage user authentication

Step 7: Deployment

- Upload to a web host with PHP & MySQL support
- Export your database using `phpMyAdmin`
- Adjust `config.php` with host credentials

---

5. Conclusion

This **Library Management System** is a simple yet functional system suitable for small libraries or academic projects. With HTML, CSS, JS, PHP, and MySQL, it can be extended with features like email notifications, overdue tracking, and user dashboards.