UMU Online Library System - Technical Documentation

Version: 1.0

Author:

Date: 18th/07/2025

---

1. System Overview

The UMU Online Library System is a dynamic web application built with PHP, CSS, and JavaScript, powered by a MySQL database. It provides students and administrators of Uganda Martyrs University with a platform to manage and access digital study materials and books online.

The system features a dual-role architecture (Student and Admin) with distinct functionalities and views tailored to each role. It is designed to be a centralized, secure, and user-friendly portal for academic resources, accessible from any modern web browser.

1.1. Core Objectives

- To provide students with 24/7 access to a wide range of academic materials.
- To centralize the management of digital resources for university staff.
- To automate the process of borrowing and returning digital items.
- To offer a modern, responsive, and intuitive user experience for all users.

1.2. Key Features

- **Role-Based Access Control:**
    - **Student View:** Access to browse and borrow materials, and manage personal account details.
    - **Administrator View:** A comprehensive backend panel for full system management.
- **Dynamic Content Management:** All materials (books, PDFs, videos, etc.) are managed through the database, allowing for easy updates without changing any code.
- **Student Account Management:** Students can self-register (restricted to a valid university email domain), log in, edit their profile information (including password), and view their borrowing history.

- **Comprehensive Admin Panel:** A secure backend for administrators to perform full CRUD (Create, Read, Update, Delete) operations on both library materials and user accounts.
- **Interactive UI/UX:**
    - Modern, responsive design that adapts to desktop, tablet, and mobile devices.
    - AJAX-powered real-time filtering on content pages.
    - Modal pop-overs for all forms, providing a seamless "app-like" workflow.
- **Core Library Functions:**
    - Functionality to borrow, return, and download materials is fully implemented.
    - Real-time updates to stock levels (available_copies) for physical books.

---

2. Technology Stack & Environment

- **Backend Language:** PHP 8.x
- **Database:** MySQL (tested with MariaDB 10.4+ via XAMPP)
- **Frontend Technologies:** HTML5, CSS3, Vanilla JavaScript (ES6+)
- **Database Interface: PDO (PHP Data Objects)** is used exclusively for all database interactions. This provides a consistent, secure interface that helps prevent SQL injection vulnerabilities.
- **Development Environment:** XAMPP Version 3.3.0 or higher (Apache, MySQL, PHP).

---

3. Installation and Setup Guide

Follow these steps precisely to set up the system on a local development machine.

3.1. Prerequisites

- A functioning local web server environment (e.g., **XAMPP**).
- Access to a MySQL database management tool (e.g., **phpMyAdmin**).
- A modern web browser (e.g., Google Chrome, Mozilla Firefox).

3.2. Installation Steps

1. **Project Files:**
    - Download or clone the project repository.
    - Place the entire project folder, named online-library, inside your web server's root directory. For XAMPP, this is typically C:/xampp/htdocs/. The final path should be C:/xampp/htdocs/online-library/.
2. **Server Services:**
    - Open the XAMPP Control Panel.
    - Ensure the **Apache** and **MySQL** services are running.
3. **Database Creation:**
    - Navigate to http://localhost/phpmyadmin/ in your browser.
    - Click the **"SQL"** tab at the top of the main page.
    - Copy the **entire content** of the provided library_db.sql script and paste it into the SQL query box.
    - Click the **"Go"** button. This script will automatically create the online_library_db database, all necessary tables with their relationships, and insert the default administrator account.
4. **Application Configuration:**
    - Navigate to the includes/ directory within your project and open the config.php file in a code editor.
    - Verify the following database credentials. **For a standard XAMPP setup with the custom 'Library' user, these are the correct defaults.**

Generated php

```
    $db_host = "localhost";

$db_user = "Library";

$db_pass = "";

$db_name = "online_library_db";

$db_port = "3308"; // IMPORTANT: Adjust if your MySQL runs on a different port.
```

    - Confirm that the BASE_URL constant correctly points to your project folder:

Generated php

```
define('BASE_URL', 'http://localhost/online-library');
```

IGNORE_WHEN_COPYING_START

content_copy download

Use code with caution. PHP

IGNORE_WHEN_COPYING_END

5. **Access and Test:**
    - Open your browser and navigate to the BASE_URL: **http://localhost/online-library/**.
    - The system homepage should load with all styles and images.

---

4. System Architecture & File Structure

The project is organized into a modular structure to enforce separation of concerns, making the codebase cleaner and easier to maintain.

Generated code

```
online-library/
├── admin/              # Admin-only pages and logic
│   ├── assets/         # Admin-specific CSS and JS
│   ├── handlers/       # PHP scripts that process admin forms (add/update/delete)
│   ├── templates/      # Reusable admin header
│   ├── dashboard.php   # The main admin dashboard
│   ├── manage_books.php  # Interface to manage library materials
│   └── manage_users.php  # Interface to manage user accounts
│
├── assets/             # Public-facing assets
│   ├── css/            # Main and login stylesheets
│   ├── js/             # Main JavaScript files
```

```
│   └── images/          # Static images (logos, backgrounds)
│
├── auth/                # Authentication logic
│   ├── handlers/        # Login/signup form processing
│   ├── login.php
│   └── signup.php
│
├── handlers/            # General-purpose backend handlers (e.g., AJAX filtering)
│
├── includes/            # Core backend files shared across the application
│   ├── templates/       # Reusable public header and footer
│   ├── config.php       # (CRITICAL) Database connection and site-wide constants
│   └── functions.php    # Global helper functions (isLoggedIn(), sanitize(), etc.)
│
├── student/             # Student-only pages and logic
│   ├── handlers/        # Scripts for borrow, return, profile updates
│   └── dashboard.php
│
└── uploads/             # Directory for user-uploaded content
    ├── book_covers/     # Stores cover images for materials
    └── study_materials/ # Stores downloadable files (PDFs, PPTs, etc.)
```

IGNORE_WHEN_COPYING_START

content_copy download

Use code with caution.

IGNORE_WHEN_COPYING_END

---

5. Database Schema

The system's data is organized into four main relational tables.

- **users**: Stores information for all system users.
    - **Fields:** id, full_name, email, student_number, password (hashed), course, role (enum('student', 'admin')).
- **materials**: A unified table for all library items, both physical (books) and digital.
    - **Fields:** id, title, author_lecturer, description, material_type (enum), genre_course, cover_image, file_path, total_copies, available_copies.
- **borrowings**: Tracks every borrowing transaction, creating a historical log.
    - **Fields:** Links user_id and material_id. Stores borrow_date, due_date, return_date, and status (enum('borrowed', 'returned', 'overdue')).
- **notifications**: A table for system-generated messages sent to users (e.g., confirmation of borrowing).
    - **Fields:** Links user_id. Stores message and is_read status.

Data Integrity: Foreign key constraints are established with ON DELETE CASCADE. This is an important feature that ensures if a user or material is deleted, all associated records (like borrowings and notifications) are automatically removed, preventing orphaned data in the database.

---

6. Default Administrator Credentials

For initial setup and testing, a default administrator account is created by the SQL script.

- **Email / Login Identifier:** admin@umu.ac.ug
- **Password:** password123

Security Note: It is highly recommended that the administrator changes this default password immediately after their first successful login. This can be done via the "Manage Users" page in the admin panel.

---

7. Future Maintenance & Customization

- **Adding Approved Courses:** The dropdown list of courses on the student dashboard is controlled by a PHP array. To add or remove courses, modify the

$approved_courses array in both student/dashboard.php and student/handlers/profile_handler.php.

- **Adjusting Pagination:** The number of items displayed per page on the "Books" and "Study Materials" pages is controlled by the $items_per_page variable at the top of handlers/filter_handler.php.
- **System Backups:** For a production environment, it is critical to perform regular backups of two locations:
    1. The uploads/ directory (which contains all user-uploaded files).
    2. The online_library_db database (via a SQL dump from phpMyAdmin or a command-line tool).