Content Management System in PHP

Documentation

By: Khalid

Date: 08/01/2024

1. Introduction

1.1 Purpose

The purpose of this CMS project is to provide a user-friendly and secure content management system for website administrators to efficiently manage and organize content.

1.2 Scope

The CMS will include

- ➔ user management (user registration, user login, updating and deleting a user),
- ➔ content creation (creating posts, updating and deleting user posts),
- ➔ frontend interface (displaying user posts),
- ➔ database management (MySql),
- ➔ security features (password hash), and
- ➔ search functionality (search post, search user).

1.3 Definitions, Acronyms, and Abbreviations

CMS: Content Management System

PHP: Hypertext Preprocessor

HTML: HyperText Markup Language

XAMPP: Cross-Platform, Apache, MySQL, PHP, and Perl

1.4 References

PHP Documentation: www.php.net/manual

HTML Documentation: www.w3.org/TR/html52/

XAMPP Documentation: www.apachefriends.org/docs

1.5 Overview

This SRS document outlines the requirements for the development of a CMS using PHP and HTML with XAMPP as the localhost server.

This documentation is prepared to elaborate how the SDLC(Software Development Life-Cycle) looks like by picking a project (CMS).

This document will contain the essential SRS - Software Requirement and Specifications document inside.

2. Overall Description

2.1 Product Perspective

The CMS will be a standalone system with an admin interface for content management and a frontend for end-users. It will interact with the MySQL database for data storage.

2.2 Product Features

The main features include

- ➔ user management,
- ➔ content creation and management,
- ➔ responsive frontend design, and
- ➔ secure database interactions.

2.3 User Classes and Characteristics

Administrators: Users responsible for managing and creating content.

End-users: Visitors accessing the website content.

2.4 Operating Environment

The CMS will run on a server configured with XAMPP, supporting PHP 7.x and MySQL 5.x.

2.5 Design and Implementation Constraints

The CMS will be designed using HTML and some JavaScript for frontend and PHP for backend development. It will utilize XAMPP for local development and testing.

2.6 Assumptions and Dependencies

The assumption is that users have basic knowledge of HTML and PHP. Dependencies include the availability of XAMPP for local development.

3. System Features

3.1 User Management

3.1.1 User Registration

Administrators can register new users.

Users provide a unique username, email, and password during registration.

3.1.2 User Authentication

Users must authenticate using their username and password.

Passwords will be securely hashed before storage.

3.1.3 User Authorization Levels

Administrators have elevated privileges.

Regular users have standard access levels.

3.2 Content Management

3.2.1 Create, Edit, and Delete Content

Administrators can create, edit, and delete articles and users.

Content includes text, images, and other media.

3.3 Frontend Interface

3.3.1 Responsive HTML Design

The frontend will have a responsive design for various devices.

HTML templates will be used for consistent styling.

3.3.2 Navigation Menus

Intuitive navigation menus for easy user interaction.

Menus will dynamically reflect content changes.

3.4 Database Management

3.4.1 Database Schema for Content Management System (CMS)

- **Users Table**

  The Users table is responsible for managing user information within the CMS. Each user is uniquely identified by a user ID.

  user_id : An auto-incremented primary key representing the unique identifier for each user.

  username : A unique string representing the user's username.

  email : A unique email address associated with the user.

  password : A secure hashed representation of the user's password.

  user_role : An enumeration ('admin', 'user') indicating the user's role, with a default value of 'user'.

- **Categories Table**

  The Categories table organizes content by providing a system of categorization.

  category_id : An auto-incremented primary key representing the unique identifier for each category.

  category_name : A unique string representing the name of the category.

- **Content Table**

  The Content table is the central repository for articles and pages within the CMS.

  content_id : An auto-incremented primary key representing the unique identifier for each piece of content.

  content_title : A string representing the title of the content.

  content_body : A text field containing the main content of the article or page.

  category_id : A foreign key referencing the Categories table, establishing a link to the relevant category.

  user_id : A foreign key linking to the Users table, indicating the author of the content.

  created_at : A timestamp indicating the creation date and time of the content.

We will establish relationships between tables for efficient data retrieval. And there will be Data Backup and Restore functionality for administrators to backup and restore data.

3.5 Security

3.5.1 Password Encryption

- The PHP “`password_hash()`” function is utilized to integrate bcrypt into the system.
- To enhance security, a unique salt is generated for each user. This salt is a random value that is combined with the user's password before hashing.

- The PHP "**`random_bytes()`**" function is utilized to generate a cryptographically secure random salt.
- The resulting hashed password is then stored in the database along with the corresponding salt.
- During the login process, the stored salt is retrieved from the database based on the username.
- The entered password is then combined with the retrieved salt, and the result is hashed using bcrypt.
- This newly hashed password is compared with the stored hashed password to authenticate the user.

3.6 Search Functionality

3.6.1 Basic Search

Users can perform basic keyword searches on content.

4. External Interface Requirements

4.1 User Interfaces

Admin Interface: HTML templates with PHP backend for content management.

Frontend: Responsive HTML templates for end-users.

4.2 Hardware Interfaces

Compatible with standard web server hardware.

4.3 Software Interfaces

XAMPP stack with Apache, PHP, MySQL.

Compatible with modern web browsers.

4.4 Communication Interfaces

System Utilizes HTTP/HTTPS for communication between clients and servers.