**Project Report: TASK MANAGEMENT SYSTEM**

**Capital University of Science and Technology**  
**Department of Software Engineering**

Introduction to Database Systems  
Ms. Sabahat Asad

**Submitted By:**

Zainab Fatima Bse233083

Meerab Arshad Bse233054

**Table of Contents**

 **Overview of the Task Management System\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

 **Project Requirements\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Functional Requirements\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Non-Functional Requirements\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 **Entities, Attributes, and Relationships\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* List of Entities and Attributes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Relationships and Cardinalities\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 **Entity Relationship Diagram (ERD)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

 **Relational Database Model (RDM)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Transformation of ERD into RDM\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 **Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* SQL Table Creation Scripts\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Insertion of Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Testing\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 **Tools and Techniques\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Development Tools Used\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Software and Platforms\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 **Implementation details**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 **Screenshots and Demonstrations\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

 **GitHub Link\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

** References \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROJECT REQUIREMENTS AND OVERVIEW**

The goal of this project is to design and implement a comprehensive task management system that organizes tasks under different projects, tracks deadlines, and categorizes tasks based on their types. Users will be able to collaborate on projects with different levels of access and roles. The system aims to improve productivity and streamline project management through intuitive interfaces and relational database management.

**Functional Requirements:**

* User registration and login functionality
* Password hashing for secure storage
* Admin and regular user roles
* Create, read, update, and delete (CRUD) operations for projects and tasks
* Assign tasks to specific users
* Define and manage task priorities (Low, Medium, High)
* Associate tasks with specific categories (Task Types)
* Link tasks to relevant projects
* Track task deadlines
* Manage user participation across multiple projects (many-to-many relationship)
* Generate reports on project progress

**Non-Functional Requirements:**

* Secure database storage using MySQL
* Scalable to handle multiple users and projects concurrently
* User-friendly interface for efficient task tracking
* Responsive system performance with minimal downtime

**Entities, Attributes and Relationships**

**Project: -**

* **projectID** (Primary Key): Unique identifier for each project
* **name**: Project name
* **deadline**: Project completion date

**Task**: -

* **taskID** (Primary Key): Unique identifier for each task
* **title**: Task title
* **description**: Detailed task description
* **deadline**: Task completion date
* **priority**: Task urgency (Low, Medium, High)
* **projectID** (Foreign Key): Links to a project
* **taskTypeID** (Foreign Key): Links to task type

**User: -**

* **userID** (Primary Key): Unique identifier for each user
* **username**: User login name
* **password**: Secure password
* **role**: User role (Admin, User)

**TaskType**: -

* **typeID** (Primary Key): Unique identifier for task type
* **typeName**: Task category name

**User\_Project** (Associative Table): -

* **userID** (Foreign Key): Links to user
* **projectID** (Foreign Key): Links to project

**Relationships and Cardinalities:**

* **Project and Task**: One-to-Many
  + Each project can have multiple tasks, but each task is linked to only one project.
* **User and Project**: Many-to-Many
  + A user can participate in multiple projects, and each project can involve multiple users. This relationship is managed through the associative table UserProject.
* **Task and TaskType**: One-to-Many
  + Each task belongs to one task type, but each task type can be associated with multiple tasks.

**Entity-Relationship Diagram: -**

A diagram of a project

Description automatically generated

**Relational Database Model: -**

A diagram of a project

Description automatically generated

**CODE:**

**Creating tables:**

-- TaskType Table

CREATE TABLE TaskType (

taskTypeID INT AUTO\_INCREMENT PRIMARY KEY,

typeName VARCHAR(50) NOT NULL

);

-- Project Table

CREATE TABLE Project (

projectID INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

deadline DATE

);

-- User Table

CREATE TABLE `User` (

userID INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL,

password VARCHAR(255) NOT NULL,

role ENUM('Admin', 'User') NOT NULL

);

-- User\_Project Table (Many-to-Many Relationship)

CREATE TABLE User\_Project (

userID INT,

projectID INT,

PRIMARY KEY (userID, projectID),

FOREIGN KEY (userID) REFERENCES `User`(userID),

FOREIGN KEY (projectID) REFERENCES Project(projectID)

);

-- Task Table

CREATE TABLE Task (

taskID INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

description TEXT,

deadline DATE,

priority ENUM('Low', 'Medium', 'High'),

projectID INT,

taskTypeID INT,

FOREIGN KEY (projectID) REFERENCES Project(projectID),

FOREIGN KEY (taskTypeID) REFERENCES TaskType(taskTypeID)

);

**Inserting sample data:**

-- Insert Task Types

INSERT INTO TaskType (typeName) VALUES ('Bug Fix'), ('Feature Development'), ('Testing');

-- Insert Projects

INSERT INTO Project (name, deadline) VALUES ('Project A', '2025-01-15'), ('Project B', '2025-02-10');

-- Insert Users

INSERT INTO `User` (username, password, role) VALUES ('admin1', 'adminpass123', 'Admin'), ('user1', 'userpass123', 'User');

-- Assign Users to Projects

INSERT INTO User\_Project (userID, projectID) VALUES (1, 1), (1, 2), (2, 1);

-- Insert Tasks

INSERT INTO Task (title, description, deadline, priority, projectID, taskTypeID)

VALUES

('Fix Login Bug', 'Resolve the bug causing login failures.', '2025-01-10', 'High', 1, 1),

('Develop New Feature', 'Implement feature XYZ for client.', '2025-01-20', 'Medium', 1, 2),

('Test Payment Integration', 'Ensure payment system is working.', '2025-02-05', 'High', 2, 3);

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

As part of the implementation process, the database schema for the task management system was successfully created, and all tables were populated with sample data. Various SQL queries were executed to test the relationships between tables, including tasks assigned to projects, users associated with multiple projects, and task types. The results confirmed that the foreign key constraints were correctly enforced, ensuring data integrity and consistency. Additionally, queries were tested to retrieve project-specific tasks and user assignments, all of which produced the expected outputs. No errors were encountered during the testing phase, and the system operates as intended, demonstrating the successful implementation of the relational database model.

**Tools and Techniques**

The implementation of the **Task Management System** leveraged a combination of robust tools and techniques to ensure efficient development, functionality, and security. The system was developed using **XAMPP**, which provided an all-in-one environment for Apache (web server), MySQL (database), and PHP (server-side scripting). **phpMyAdmin** was utilized to manage the MySQL database, including creating tables, managing relationships, and executing queries. For coding, **Visual Studio Code** and **Notepad++** were employed, offering features like syntax highlighting and debugging to streamline the development process. Testing and debugging were conducted using browsers like Google Chrome and Microsoft Edge to ensure compatibility and responsiveness.

On the technical side, the system’s front end was built with **HTML** for structuring web pages and **CSS** for a visually appealing, pink-themed design, ensuring an intuitive user interface. The back end, powered by **PHP**, handled functionalities like user authentication, task creation, editing, and deletion. Security was a key focus, with password hashing implemented using password hash and password verify to store credentials securely, while prepared statements were used in SQL queries to prevent SQL injection attacks. The database was normalized to maintain data integrity and efficiency, with clearly defined relationships between users, tasks, projects, and task types.

Testing was conducted rigorously to validate system features such as task filtering, role-based access control, and error handling. The project was designed to be scalable and adaptable, with responsive design ensuring usability across devices. This combination of tools and techniques resulted in a secure, functional, and user-friendly system suitable for task management needs.

**Implementation details: -**

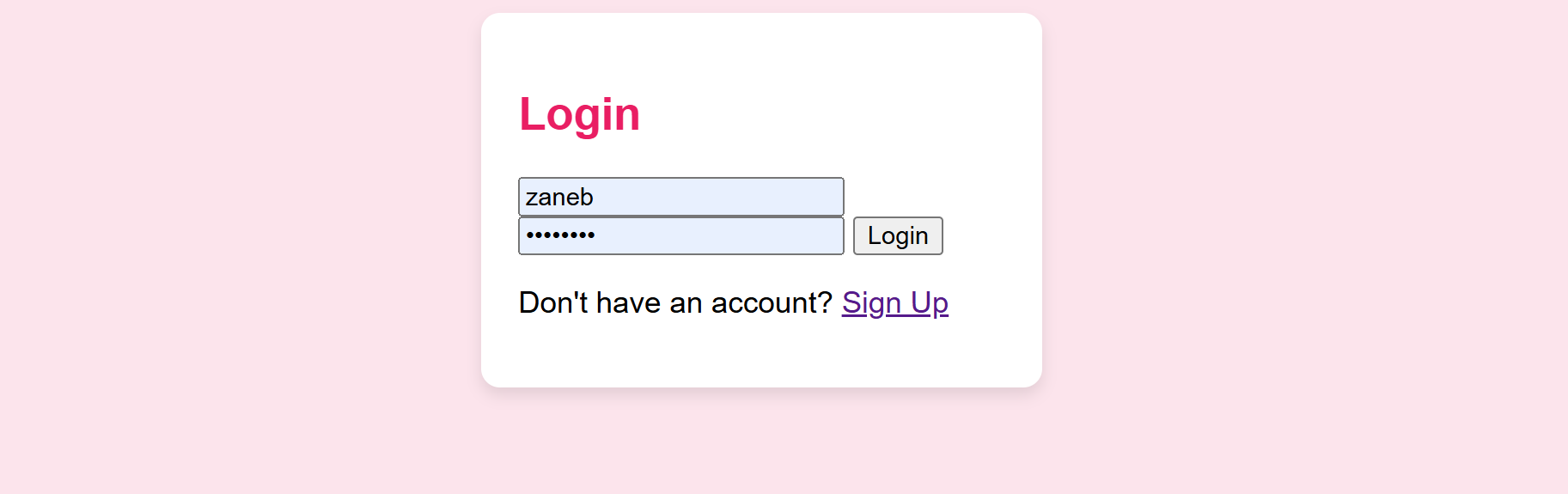
The **Task Management System** is a web-based application built using PHP, MySQL, HTML, and CSS, designed to streamline task organization and management. It incorporates role-based access control, allowing Admins to create, edit, and delete tasks, while regular users can only view them. Tasks are associated with projects, categorized by type, and assigned priorities (Low, Medium, High) with deadlines to ensure efficient tracking. The system features a user-friendly, pink-themed interface, making navigation intuitive and visually appealing.

The backend is powered by PHP, using prepared statements to ensure secure interactions with the MySQL database. User authentication employs hashed passwords for security, leveraging password hash to store credentials securely. The database consists of well-defined tables for users, tasks, projects, and task types, ensuring a normalized structure that maintains data integrity. Dynamic filtering on the dashboard allows users to filter tasks by priority, enhancing usability and adaptability for various use cases.

The application is implemented on XAMPP, enabling local hosting with Apache and MySQL. The frontend is styled with CSS to deliver a consistent, responsive interface. Error handling is integrated into key operations, ensuring clear feedback for the user in case of invalid inputs or failed database actions. This system has been rigorously tested to ensure compatibility across modern browsers and efficient handling of user operations. It is ideal for individuals or teams seeking a robust task management solution.

**Images or screenshots: -**

**Login:**



**Signup:**

A screenshot of a login form

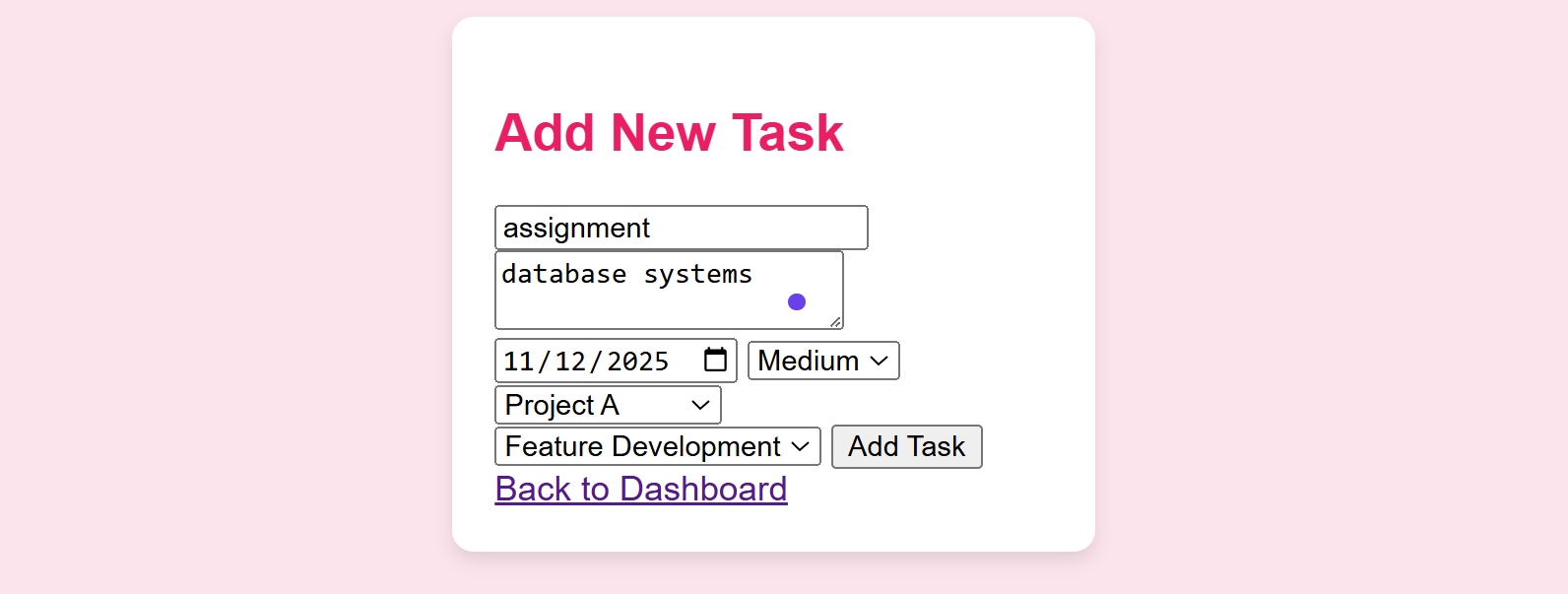
Description automatically generated

**Dashboard:**

A screenshot of a computer screen

Description automatically generated

**Adding tasks:**

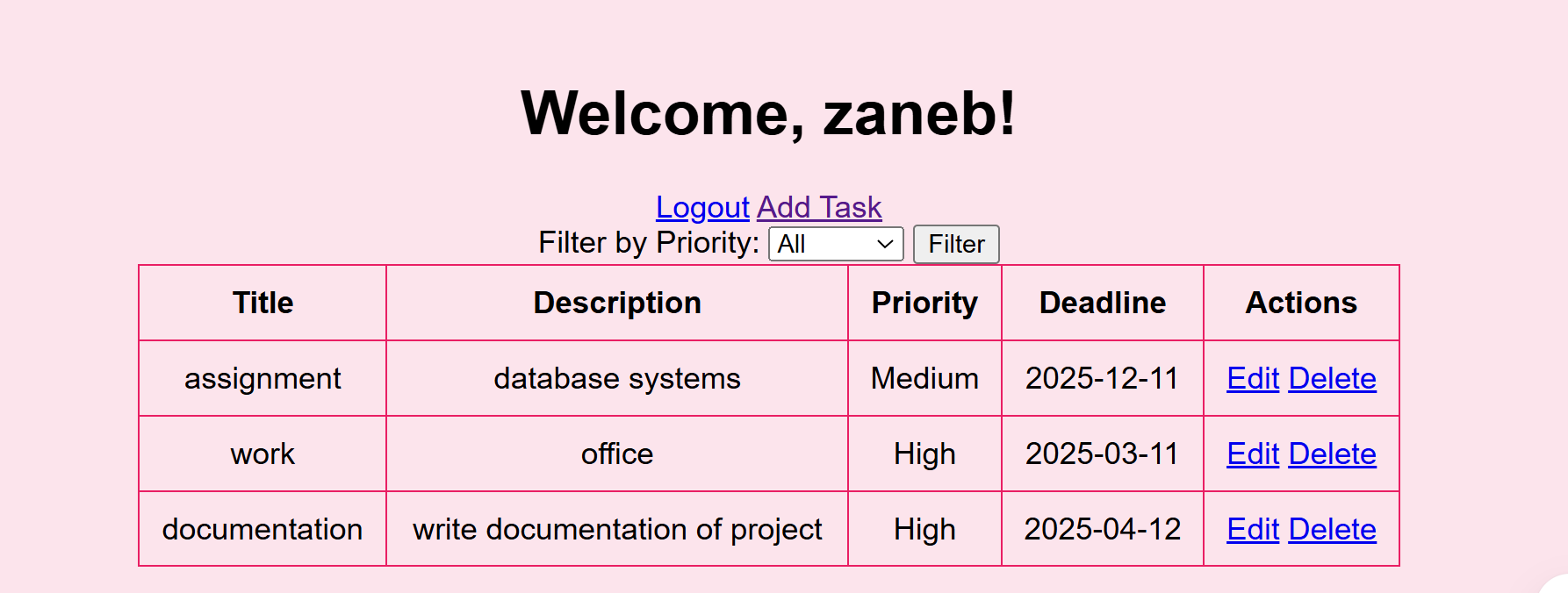
****

**Filter by priority check:**

A screenshot of a computer

Description automatically generated

**Delete task and edit check:**



**Final image of dashboard:**

A screenshot of a computer program

Description automatically generated

**GitHub Link: -**

**References**

1. **XAMPP Official Documentation**  
   Apache Friends. (n.d.). XAMPP Documentation. Retrieved from <https://www.apachefriends.org/>
2. **PHP Manual**  
   PHP Group. (n.d.). PHP Documentation. Retrieved from <https://www.php.net/docs.php>
3. **MySQL Documentation**  
   Oracle. (n.d.). MySQL Documentation. Retrieved from <https://dev.mysql.com/doc/>
4. **phpMyAdmin Documentation**  
   phpMyAdmin Team. (n.d.). phpMyAdmin Documentation. Retrieved from <https://www.phpmyadmin.net/docs/>
5. **HTML & CSS Basics**  
   W3Schools. (n.d.). HTML and CSS Tutorials. Retrieved from <https://www.w3schools.com/>