STUDENT MANAGEMENT SYSTEM



NAME: “STUDENT NAME”

SCHOOL: “STUDENT SCHOOL”

ADM NO: “STUDENT ADMISSION NUMBER”

PROJECT TITLE: STUDENT MANAGEMENT SYSTEM

COURSE TITLE “STUDENT COURSE”

DATE: “AUGUST 2024”

# ABSTRACT

The student management system was made and designed specifically to cater for all student functionalities. This include functions such as registering new students, updating student details, updating student academics, sending notifications to students, mark student attendance and viewing all student related reports. The student management is designed using HTML, CSS, JAVASCRIPT, PHP and MYSQL. All these key components have been integrated in creating a well-functional and responsive web application.

This system has been developed with a well-defined user-authentication system that will only allow teachers with admin privileges to manage student related information. The system also has a code in place to handle password resetting incase the user, has forgotten their password. The system has been developed through a process of requirements gathering, system analysis, design analysis, testing and deployment. This method of design ensures that the system can easily be scaled and expanded if need arises in the near future.

The main-difference between the modernized school management system and the manual system is that, this system enables the school to save on more time that would be otherwise be used in benefiting the students. The system will however require the school to invest in standard computers, furniture and a staff training to learn how to use, manage and troubleshoot the student management system.

This comprehensive documentation provides a guide of how the student management system was made, how it is going to be implemented and how it will automate and make running student related details within the academic institution easier and more efficient. The primary objective of this system is enhance task management, reduce manual error and provide access to accurate student data.

This documentation also explains the evaluation on the system performance and how future upgrades maybe made possible. The student management system will serve as a useful tool that will both benefit the teachers and the students. As a result, the student will receive better services provided by the school and the teachers will be able to serve the students in the best way they can.

Table of Contents

[ABSTRACT 2](#_Toc175220051)

[CHAPTER 1: INTRODUCTION 5](#_Toc175220052)

[1.1 BACKGROUND 5](#_Toc175220053)

[1.2 PROBLEM STATEMENT 6](#_Toc175220054)

[1.3 OBJECTIVES 8](#_Toc175220055)

[1.4 SCOPE 9](#_Toc175220056)

[Included in Scope 9](#_Toc175220057)

[Excluded from the Scope 10](#_Toc175220058)

[CHAPTER 2: SYSTEM ANALYSIS 11](#_Toc175220059)

[2.1 REQUIREMENT GATHERING 11](#_Toc175220060)

[1. INTERVIEWS 12](#_Toc175220061)

[2. QUESTIONNAIRES 12](#_Toc175220062)

[3. DOCUMENT ANALYSIS 13](#_Toc175220063)

[CASE DEVELOPMENT 13](#_Toc175220064)

[REQUIREMENT SPECIFICATIONS 13](#_Toc175220065)

[VALIDATIONS 13](#_Toc175220066)

[2.2 SYSTEM DESCRIPTION 14](#_Toc175220067)

[2.2.1 REVIEW OF THE SYSTEM 14](#_Toc175220068)

[2.2.2 SYSTEM ARCHITECTURE 14](#_Toc175220069)

[2.2.3 CORE MODULES 15](#_Toc175220070)

[2.2.4 SYSTEM FEATURES 16](#_Toc175220071)

[2.2.5 SYSTEM LIMITATIONS 16](#_Toc175220072)

[2.3 USE CASE DIAGRAMS 16](#_Toc175220073)

[2.3.1 TEACHERS 16](#_Toc175220074)

[2.3.2 STUDENTS 17](#_Toc175220075)

[2.3.3 PARENT 18](#_Toc175220076)

[2.3.4 ADMINISTRATION 18](#_Toc175220077)

[2.3.5 STUDENT MANAGEMENT SYSTEM 19](#_Toc175220078)

[CHAPTER 3: SYSTEM DESIGN 20](#_Toc175220079)

[3.1 DATABASE DESIGN 20](#_Toc175220080)

[3.1.1 DATABASE STRUCTURE 20](#_Toc175220081)

[3.1.2 DATA INTEGRITY AND CONSTRAINTS 21](#_Toc175220082)

[3.2 SYSTEM ARTITECTURE 22](#_Toc175220083)

[3.3 USER INTERFACE DESIGN 22](#_Toc175220084)

[3.3.1 LAYOUT AND NAVIGATION 22](#_Toc175220085)

[CHAPTER 4: IMPLEMENTATION 23](#_Toc175220086)

[4.1 DEVELOPMENT ENVIRONMENT 23](#_Toc175220087)

[4.1.1 HARDWARE REQUIREMENTS 23](#_Toc175220088)

[4.1.2 SOFTWARE REQUIREMENTS 24](#_Toc175220089)

[4.1.3 CONFIGURATION MANAGEMENT 25](#_Toc175220090)

[4.1.4 DOCUMENTATION TOOLS 25](#_Toc175220091)

[4.2 CODING STANDARDS 25](#_Toc175220092)

[4.2.1 GENERAL CODING PRACTICES 25](#_Toc175220093)

[4.3 PHP IMPLEMENTATION 26](#_Toc175220094)

[4.3.1 db\_connect.php 27](#_Toc175220095)

[4.3.2 SECURITY MEASURES 27](#_Toc175220096)

[4.3.3 ERROR HANDLING AND LOGGING 27](#_Toc175220097)

[4.4 HTML/CSS IMPLEMENTATION 28](#_Toc175220098)

[4.4.1 HTML STRCUTURE 28](#_Toc175220099)

[4.4.2 CSS STRUCTURE 28](#_Toc175220100)

[4.5 JAVASCRIPT IMPLEMENTATION 29](#_Toc175220101)

[CHAPTER 5: TESTING AND EVALUATION 30](#_Toc175220102)

[5.1TESTING STRATEGY 30](#_Toc175220103)

[5.1.1 TESTING OBJECTIVES 30](#_Toc175220104)

[5.1.2 TESTING METHODOLOGIES 31](#_Toc175220105)

[5.1.3 TESTING PROCESS 31](#_Toc175220106)

[5.1.4 TEST COVERAGE 32](#_Toc175220107)

[5.1.5 TEST ENVIRONMENT 32](#_Toc175220108)

[5.2 TEST CASES 32](#_Toc175220109)

[5.2.1 TEST CASE STRUCTURE 32](#_Toc175220110)

[5.2.2 SAMPLE TEST CASES 33](#_Toc175220111)

[5.3 EVALUATION 35](#_Toc175220112)

[CHAPTER 6: DEPLOYMENT 36](#_Toc175220113)

[6.1 DEPLOYMENT PROCESS 36](#_Toc175220114)

[6.2 MAINTENANCE 38](#_Toc175220115)

[CHAPTER 7: CONCLUSION AND FUTURE WORK 39](#_Toc175220116)

[7.1 SUMMARY 39](#_Toc175220117)

[7.2 FUTURE ENHANCEMENTS 40](#_Toc175220118)

[REFERENCES 41](#_Toc175220119)

[APPENDICES 42](#_Toc175220120)

# CHAPTER 1: INTRODUCTION

## BACKGROUND

Managing student’s information in our day has been critical in ensuring smooth operation of educational institutions. Previously, student records, attendance grades and other essential data were maintained through manual processes which mostly led to errors and difficulties in retrieving data. With the increasing student populations, a need for a more streamlined and more efficient system to manage information is needed.

Digital technology has changes the way educational institutions operate by providing tools that enhance administrative functions and also improve overall educations experience for both the teachers and the students. A computerized student management system is a solution that addresses the challenges faced by the institution in managing student related data. This system will offer a centralized platform for storing, managing and retrieving student information hence reducing paper-trails and minimizing potential errors.

The student management system is developed so as to provide an integrated and efficient solution that will serve all the needs of educational institutions. Using PHP for server-side processing, MYSQL for database management and HTML and CSS for front-end and JavaScript for back-end, the student management system is made to be user-friendly so as to manage all student related data. The system supports features such as attendance-tracking, registration, real-time notification and performance analytics among others.

The student management system is developed for a modern school system where processing of information and decision making is made with ease and shortest time possible. The manual system brought up the issue of redundancy, poor data storage and lack of data privacy. The implementation of a new computerized system will solve the issue of data storage by using a digital database system

made possible by MYSQL.

## 1.2 PROBLEM STATEMENT

The introduction of the computerized system aims to solve problems posed by the current manual system. The current manual student management system posed the following problems:

1. There was a lack of a centralized management system – The manual system had form a centralized management system that would aid in easier decision making and processing of information
2. There was lack of data privacy – Data was stored in manual filling cabinets that posed no means of security. Financial ledgers used could easily be lost and no means of accounting for the data was lost.
3. Redundancy – Duplicate student, teacher and academic records could be kept within the manual filling cabinet. This is could cause a massive waste of space and could easily lead to loss of data.
4. Waste of time in retrieving information – a lot of time is wasted in trying to retrieve data, as data is stored manually in file cabinets, and accessed by manually sorting through the index of the files.
5. There is no backup of information – creating a backup for a manual student management system would otherwise be expensive and time-consuming hence when data is lost within the system it is forever lost.
6. It is expensive to maintain – A manual management system would require file-cabinets and ledgers. Updating this system would otherwise require new equipment’s that would otherwise be expensive.
7. It is not-environment friendly – The manual management system mainly stores their data in files and ledgers that uses paper as a means of data entry. Most of these papers would otherwise be thrown out when no longer used hence polluting the school environment.
8. Information can easily be compromised – Information stored in manual ledgers and filling cabinets can easily be compromised by either human factors such as poor storage or environmental factors such as poor weather conditions.
9. Information cannot be easily retrieved – Information takes time to retrieve from the filing cabinets, sometimes files may not be properly sorted and indexed hence more time would be taken in getting student record.
10. Little information can be stored – the manual filling method used in the management was limited due to space, if more information was to be stored in the system, then more offices and storage space would otherwise need to be constructed hence the system would become even more expensive to maintain.

The above problems are some of the limitations that the manual student management system faced. The computerized student management system aims to solve the problems above and give the school administration and easier time in managing student related information.

## OBJECTIVES

The computerized system of the student management system was launch with the aim of achieving goals that the manual system failed to deliver. Some of the objectives of the new student management system will include:

1. To enhance security and privacy – The current system will improve on security by having a user authentication system that will authenticate all system users before accessing the student information.
2. To improve access to information – The system will allow for teachers and users to search throughout the system for student information using PHP as a server side language to ask for information from the database.
3. To support data-driven decision making – With access to a computerized system, more data will be accessed at a glance making it easier for teachers and administration to make decisions quickly and more efficiently.
4. To enhance administration efficiency – The administration will be able to drive the school better as accurate information of the student is kept on a computerized system that can be easily accessed by the management.
5. To enhance environment responsibility – Switching to a computerized system means that less paper and less materials that are gotten by destroying our environment will be no longer needed. The school as a result will be as clean as it can possibly get.
6. To automate attendance management – The system will allow teachers to easily mark attendance and generate attendance reports, as a result less time will be used than in the manual record keeping.
7. To simplify grading and report generation – Ranking of students has never been easier than right now while using the new student management system. Formulas for calculating student grades and code for generating academic reports has made it easier for teachers to handle student academics and less time wasted during the manual grading system.
8. To enhance user experience – With the new student management system, teachers and other user will be able to interact with a nice and well-designed dashboard that has been designed to cater for the needs of all users.
9. To support online registration and enrollment – Now students will be able to enroll online at the comfort of their home without the need to come to school for enrollment. As a result, students will get more time to prepare themselves for the actual reporting day.
10. To improve resource management – With a centralized computerized system, student resources as easier to manage as more users will be held accountable due to record tracking and easier management handling.
11. To enable academic tracking – With the new student management system, teachers can actually track their students’ performance and do the necessary academic measures as needed. The previous system was tiresome for a teacher to do analysis for all their student’s performance analysis.
12. To promote integration with other academic tools. – The system is designed in such a way as to promote integration between other learning management systems tools that incorporate both the parents, teachers and the students.
13. To facilitate and enhance parental involvement – The system will allow parents to view their child’s status within the school. All academic and attendance reports can be easily to shown to the parent for the right measures to be taken.
14. To streamline communication channels – With a new computerized system, it has become much easier to communicate with teachers and students and update them with new notifications on upcoming school events.

## SCOPE

The scope of the student management system covers a wide range of functions that is meant to support academic and administrative factors within the school. The system was built to serve various users including teachers, parents, students, and the administration by providing specific features based on their permissions and roles.

The following outlines the boundaries and components of this system:

### Included in Scope

1. Academic Performance – Management of student grades by teachers with automated calculation of grades and overall performance. There will be generation of report cards by performance analytics accessed by the teacher and forwarded to the student
2. Attendance Tracking – There will be automated attendance tracking allowing teachers to mark students as either present or absent with a simple user-interface. Generation by attendance reports will be viewable by teachers and the administration.
3. Student Information Management – Student data will be stored within a centralized database management system. Access to student profile information will be accessed by only the administration and other authorized personnel’s.
4. Reporting and Analytics – There will be the generation of various reports including, attendance records, academic reports, and student enrollment reports.
5. Online registrations –Online platform for student enrollment simplifies the registration process for new and returning students.
6. Security and Privacy – Implementation of a user authentication system and data encryption to protect sensitive student information.
7. Integration - Integration with other educational platforms and tools such as a Learning Management System.
8. Notifications – Integrated messaging systems for sending notifications and alerts to teachers, parents, and students.
9. User Role management – Ensures that users only have data relevant to their role. Dashboards that are customizable based on the needs of administrators, students, teachers and parents.

### Excluded from the Scope

1. Financial Management System – The system does not manage any financial transactions such as school fees or teachers’ payroll and school budgeting.
2. Learning Management – The system does include a full learning management system for online classes or online learning.
3. Mobile Application Development – The system is limited to a web based platform and cannot be included in the development of a mobile app. However, the web interface is designed to be responsive on all devices.
4. Physical Resource Management – Management of physical resources such a library books and other resources is not included in the scope of this system
5. Advanced data analytic – The system uses basic analytic features, advanced data analytics and machine learning insights are not included in this version of the system.

# CHAPTER 2: SYSTEM ANALYSIS

## 2.1 REQUIREMENT GATHERING

The development of the student management system began with a comprehensive requirement gathering phase where different methods of gathering information was used on the teachers, parents, students and the administration. This was necessary so as to identify the problems and the needs that the system would address and how it would be implemented.

Here are some of the methods used during the gathering phase:

### INTERVIEWS

* Interviews were carried with the administration to identify how daily school tasks were handled and how they could be further improved. The areas of focus were student registration, attendance tracking and report generation.
* Teachers were interviewed to determine their needs in managing student records, grading and in attendance tracking. Their needs were used to determine how a user-interface will be integrated with the current educational tools.
* Student provided their wants during the interview such that the system should allow for students to track their academic records and it should update them on upcoming notifications.
* Parents were interviewed and they wanted the system to give them real-time updates based on the student performance and attendance records.

### QUESTIONNAIRES

Questionnaires were handed out to teachers, students, parents and the administration to gather data on the specific needs and preferences of each user. The questionnaire covered a range of topics including:

#### 1.FUNCTIONAL REQUIREMENTS

Essential features that all users considered useful such as enrollment, attendance tracking and grade management

#### 2.SECURITY AND PRIVACY CONCERNS

The sensitive student information and the access to its data is expected to be in compliance with data protection and regulations.

#### 3.USABLILITY REQUIREMENTS

Preferences regarding the user interface, ease of navigation and overall user-experience.

#### 4.INTERGRATION AND COMPATIBILTY

There is a requirement for the system to integrate with the previous software used by the school such as the Learning Management System and other administrative tools.

### DOCUMENT ANALYSIS

Existing documentation such as school policy, academic regulations and school reports were reviewed to ensure that the system would comply with institutional policies and leverage the existing available resources. The analysis helped in identifying:

* Regulatory Compliance Requirements: Ensuring the system follows all educational standards and regulates as outlined by the relevant authorities.
* Technical Requirements: Ensuring the system is compatible with the facilities IT infrastructure, including all hardware, software and network specification.

### CASE DEVELOPMENT

Based on the information gathered, cases were developed to show scenarios, that the system would need to otherwise support. The scenarios provided a clear understanding of how different users would interact with the system.

### REQUIREMENT SPECIFICATIONS

The findings of the information gathering methods were compiled into a detailed requirement specification document. This document served as the foundation of the system design having the following components:

#### FUNCTIONAL REQUIREMENTS

A comprehensive list of features and functionalities that the system must provide to meet the need of its users.

#### NON-FUNCTIONAL REQUIREMENTS

Specifications related to the system performance, scalability, security and usability.

#### CONSTRAINTS AND ASSUMPTIONS

Limitations and assumptions that were considered during the development process included, budgetary constraints, timeline-considerations and resource availability.

### VALIDATIONS

The gathered requirements were based on their importance and feasibility. This was validated through follow up meetings with teachers, student and parents to ensure that the most critical needs were addressed first.

## 2.2 SYSTEM DESCRIPTION

Education institutions can use the Student Management System to manage student data and perform additional administrative tasks. Numerous modules are included in it, including ones for processing grades, monitoring attendance, registering students, and communication services. PHP is used for the application's functionality, and MySQL is used for database storage. Utilizing HTML, CSS, and JavaScript, the system was created with its interface in mind to give users a responsive and easy-to-use experience.

2.2.1 REVIEW OF THE SYSTEM   
  
The student management system aims to offer centralized control over all information and activities about students. In order to guarantee that administrators, instructors, students, and parents have prompt access to the information they require, it is intended to increase the effectiveness and accuracy of data management procedures.

### 2.2.2 SYSTEM ARCHITECTURE

The application is divided into discrete tiers by the system's multi-tier architecture:

#### PRESENTATION LAYER

The user interface (UI) and user experience (UX) are handled by this layer. Because HTML, CSS, and JavaScript were used in its development, the system is responsive and available on a variety of devices and web browsers. Users can engage with the system effectively because the user interface (UI) components are designed to be intuitive.

#### BUSINESS LOGIC LAYER

This layer, which is PHP-developed, handles the system's essential functions. This layer communicates with the database to retrieve or update data, processes user requests, and implements business rules. It also oversees error management, data validation, and authentication, making sure the system runs smoothly and safely.

#### DATA LAYER

MySQL is the relational database management system (RDBMS) that powers the data layer. It stores all the information about students, courses, attendance, grades, and user accounts. The database is made to guarantee safe data storage, support effective query processing, and preserve data integrity.

### 2.2.3 CORE MODULES

The SMS is composed of multiple main modules, each handling a particular section of student management:

#### USER MANAGEMENT

Role-based access control, login, and user registration are managed by this module. Users are given roles with specified permissions and access rights, such as administrator, teacher, student, or parent. All users are guaranteed safe authorization and authentication by the system

STUDENT INFORMATION MANAGEMENT  
Administrators can keep, modify, and access comprehensive student records using this module, which include contact information, academic background, and personal data. The system facilitates the bulk uploading of student data and guarantees that all data is maintained in an organized manner for effortless retrieval and analysis.

#### ATTENDANCE MANAGEMENT

Teachers can use this module to manage student attendance marking and recording. The attendance module will allow teachers to mark student attendance, view attendance on particular days and generate the respective reports.

#### GRADE MANAGEMENT

The grade management module is to be used to calculate student grades, ranking and present individual and general student academic reports. This module will allow teachers to input student results in a centralized database system and be able to generate accurate reports from it.

#### REGISTRATION MANAGEMENT

The registration management module is meant to manage how new students are to be registered into the school and how the existing student data can also be updated, viewed and deleted. It will also manage how new users of the student management system will be registered into the system

#### NOTIFICATION MANAGEMENT

The notification management module is meant to manage how student notifications will be sent by the teachers to the students. This ensures that all notifications sent by the user can be viewed by other teachers and the students themselves.

#### REPORT MANAGEMENT

The report management module is to be used to manage all reports to be generated related to the student management system. The module will allow teachers and students to generate accurate reports related to student academics and attendance records.

### 2.2.4 SYSTEM FEATURES

The following are the system features of the student management system:

1. Responsive design to work on all devices.
2. High levels of security are made possible by user authentication.
3. Customizable based on user needs
4. Scalable as it is built to handle a large number of students.
5. Integratable as it can be used with other learning tools such as e-learning.

### 2.2.5 SYSTEM LIMITATIONS

While web-based student management offers a lot of advantages, it also has some limitations which include:

1. It depends on internet connectivity to work – Online modules such as Bootstrap, Font Awesome and other online jQuery will require internet connectivity to work.
2. Difficulty in initial setup – The initial setup of the student management system and its installation will require technical expertise in database management and SQL that might cause a challenge to users.
3. Limited Offline access – The system does not currently support and offline mode hence users without internet connectivity will find it difficult to manage student information.

## 2.3 USE CASE DIAGRAMS

This is a visual representation of the function to be done by each users. Teachers, students, parents and the administrations will each have different functions.

### 2.3.1 TEACHERS

TEACHER

MANAGE

STUDENT

GRADES

GENERATE

REPORTS

COMMUNCATE WITH STUDENTS

AND TEACHERS

VIEW STUDENT ATTENDANCE

### 2.3.2 STUDENTS

STUDENT

VIEW

PERSONAL

DETAILS

CHECK

ATTENDANCE

VIEW

GRADES

COMMUNICATE WITH   
TEACHERS

### 2.3.3 PARENT

PARENTS

VIEW

CHILDS

ATTENDANCE

VIEW

CHILDS

ACADEMICS

COMMUNCATE WITH

TEACHERS

### 2.3.4 ADMINISTRATION

ADMINISTRATION

MANAGE

USER

ACCOUNTS

CONFIGURE SYSTEM

SETTINGS

GENERATE

REPORTS

### 2.3.5 STUDENT MANAGEMENT SYSTEM

STUDENT

MANAGEMENT SYSTEM

STUDENT

PARENT

TEACHERS

ADMINISTRATION

UPDATE

ACADEMIC

RECORDS

UPDATE

ATTENDANCE

RECORDS

GENERATE

REPORTS

VIEW

ACADEMIC

REPORT

VIEW

ATTENDANCE

REPORT

LOG-IN

MANAGE USER

ACCOUNTS

CONFIGURE SYSTEM

SETTINGS

ENTER

STUDENT

GRADES

ENTER

STUDENT

ATTENDANCE

VIEW

STUDENT

REPORT

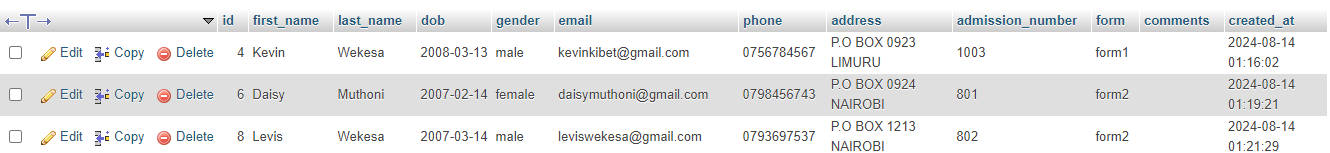
# CHAPTER 3: SYSTEM DESIGN

## DATABASE DESIGN

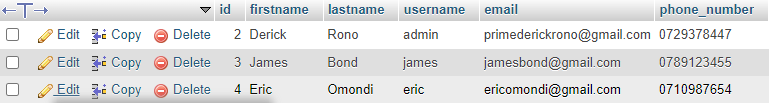
The database design is the main component of the student management system. It consists of tables where user data and student-related data are stored. The following are the tables present in the student management database system:

### 3.1.1 DATABASE STRUCTURE

#### Student Table



#### Users Table



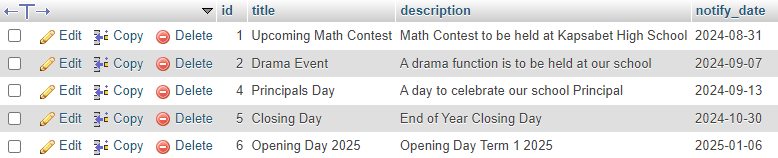
#### Academics Table



#### Attendance Table



#### Notifications Table



### DATA INTEGRITY AND CONSTRAINTS

The Student Management System database includes various constraints to maintain data integrity:

* **Primary Key Constraints:** Ensure each record is unique.
* **Foreign Key Constraints:** Maintain referential integrity between related tables.
* **Unique Constraints:** Ensure that certain fields, such as email addresses, are unique across the database.
* **Not Null Constraints:** Ensure that critical fields, like student names and grades, are always populated.

## SYSTEM ARTITECTURE

The student management system is based on a multi-tier model, separating the system into layers each with their own responsibilities:

1. **Presentation Layer:** This is the layer responsible for the user interface and user experience(UI/UX). It is developed using HTML, CSS and JavaScript to ensure a responsive and well-designed interface through which users can interact with the system.
2. **Application Layer:** This is the layer that contains the business logic of the Student Management System. It is made possible using PHP, which handles user requests and retrieves the relevant information from the database.
3. **Data Layer:** This is the layer responsible for data storage and retrieval. It is made possible using MYSQL that stores all the data related to students, attendance. Grades and notifications

## 3.3 USER INTERFACE DESIGN

The User-Interface design of the Student Management System focuses on providing a simple, user-friendly and a nice experience for all users. This ensures all users of the system can effectively use the system and make changes and updates to student related data.

The UI follows the principle of simplicity, consistency, responsiveness and accessibility during the designing phase. The Color Scheme used in the system was made to be professional using blue and white for the primary colors, light gray and black for the secondary colors and Ubuntu and san-serif for the font-family.

### LAYOUT AND NAVIGATION

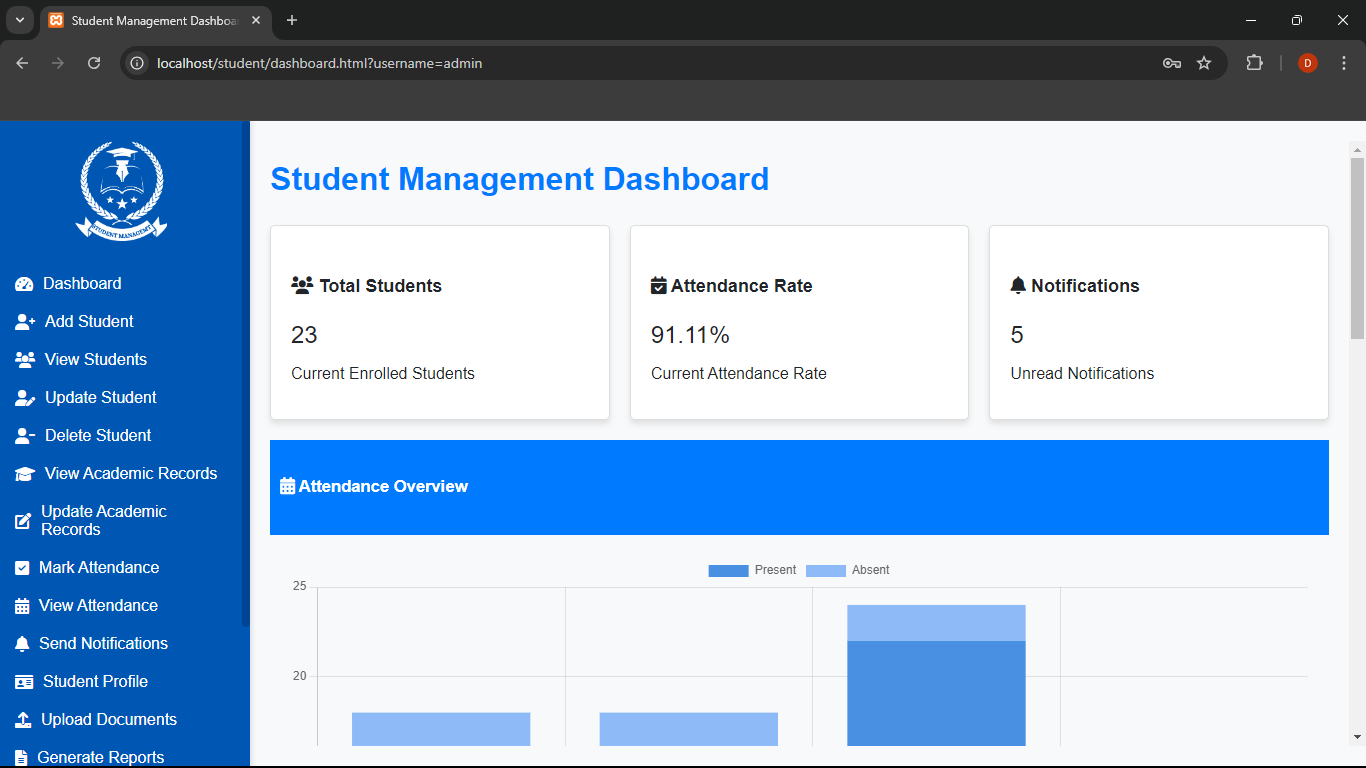
The layout of the Student Management System us designed to be simple and organized creating an easy to use navigation:

Header: The header includes the system logo and the navigation links.

Sidebar: The side bar contains all navigation links including sections for the student details. Attendance and notifications.

Main-content: Displays all content related to the navigation link that is clicked. All information concerning the link and input forms is displayed here.

Footer: Contains links related to the name of the system and the copyrighted content.



# CHAPTER 4: IMPLEMENTATION

## 4.1 DEVELOPMENT ENVIRONMENT

The development of the Student Management System was carried out in a carefully configured environment to ensure consistency, scalability, and reliability. The following tools, platforms, and configurations were used:

### 4.1.1 HARDWARE REQUIREMENTS

* **Processor:** Intel Core i5 or equivalent
* **RAM:** 8 GB or more
* **Storage:** 256 GB SSD or more
* **Operating System:** Windows 10, mac OS, or Linux (Ubuntu 20.04 or later)

### 4.1.2 SOFTWARE REQUIREMENTS

1. **Operating System:**

* Primary: Windows 10
* Secondary: Ubuntu 20.04 LTS

1. **Integrated Development Environment (IDE):**

* Visual Studio Code 1.XX (with extensions for PHP, JavaScript, HTML, and CSS)
* Sublime Text 3 (optional)

1. **Version Control System:**

* Git 2.X
* GitHub or GitLab for remote repositories

1. **Database Management System:**

* MySQL 8.0 (managed via XAMPP or WAMP)

1. **Web Server:**

* Apache 2.4 (included in XAMPP)

1. **Programming Languages:**

* PHP 8.3
* JavaScript
* HTML5
* CSS3

1. **Frameworks & Libraries:**

* Bootstrap 5 for responsive design
* jQuery 3.X for DOM manipulation and AJAX
* Chart.js for creating dynamic charts

1. **Browser:**

* Google Chrome 91 or later (for testing and debugging)
* Mozilla Firefox 89 or later (as a secondary testing environment)

### 4.1.3 CONFIGURATION MANAGEMENT

1. **Environment Configuration:**
   1. Configuration files were used to manage database connections, API keys, and other sensitive settings.
2. **Error Reporting:**
   1. Enabled detailed error reporting in development (display errors = On) and disabled in production (display errors = Off).
3. **Security Configurations:**
   1. Implemented SSL certificates for secure HTTPS communication.
   2. Enforced secure coding practices (e.g., input validation, prepared statements).
4. **Testing Environment:**
   1. PHP Unit was used for unit testing.
   2. Manual and automated browser testing using Selenium.

### 4.1.4 DOCUMENTATION TOOLS

1. Microsoft Word for writing documentation.
2. Microsoft Publisher for creating system architecture.
3. Canva for creating system Logos

## 4.2 CODING STANDARDS

To maintain a high level of code quality and consistency throughout the development of the Student Management System, the following coding standards were adopted:

### 4.2.1 GENERAL CODING PRACTICES

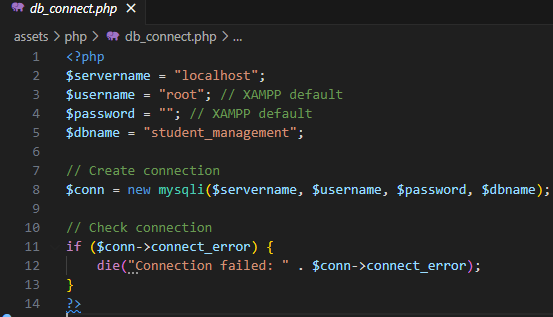
* **Code Readability:**
* Used clean, self-explanatory code with appropriate comments.
* Used meaningful variable, function, and class names that convey their purpose.
* Limited line length to 80-100 characters for better readability.
* **Indentation:**
* Used 4 spaces per indentation level (no tabs).
* Ensured consistent indentation across all files.
* **File Structure:**
* Grouped related functions and classes into appropriately named files.
* Organized code into folders such as assets/php, assets/js, and assets/css to maintain a clear directory structure.
* **Commenting:**
* Used comments to explain complex logic and decisions, not obvious code.

## 4.3 PHP IMPLEMENTATION

The Student Management system backend is powered by PHP for its server-side interaction with the MYSQL database. This section outlines the key PHP components and their implementation. The system's PHP design is modular and well-organized, with the codebase divided into distinct files and folders for easier maintenance. The principal elements consist of:  
**1. Core Files:** These are responsible for managing session management, user authentication, and database connectivity, among other essential system functions.  
**2. Controllers:** These files interact with the database to update, retrieve, and display data. They also process user input.  
**3. Views:** These HTML templates, which frequently include integrated PHP for dynamic content, are what show the data to the user.  
**4. Models:** These manage database interactions and depict the structure of the database.

### 4.3.1 db\_connect.php

This is the php file used to establish the database connection to the MYSQL database. Use It and change it to your server name and address:



### 4.3.2 SECURITY MEASURES

Security is a critical aspect of the PHP implementation, with several measures in place to protect the system:

* **Input Validation:** All user inputs are validated and sanitized to prevent SQL injection and XSS attacks.
* **Password Hashing:** User passwords are securely hashed using password hash () before storing in the database.
* **Session Management:** Secure session handling practices are implemented, including session timeout and regeneration of session IDs.

### 4.3.3 ERROR HANDLING AND LOGGING

The system includes robust error handling mechanisms to ensure smooth operation and easy debugging.

* **Error Reporting:** Enabled during development, with detailed logs stored in files for production environments.
* **Custom Error Pages:** User-friendly error pages are displayed in case of critical failures, ensuring a good user experience.

## 4.4 HTML/CSS IMPLEMENTATION

The UI of the Student Management System was designed using HTML for Structure and CSS for Styling. The following section outlines how HTML and CSS were implemented:

### 4.4.1 HTML STRCUTURE

The HTML has been designed to serve all the functions related to the student. The main HTML pages include:

1. Index.html – This is the main-page used for logging into the student management system.
2. Dashboard.html – Displays the student details summary, attendance, academics and all relevant reports.
3. Mark-attendance.html – Used to mark student attendances.
4. Send-notifications.html – Used to send student notifications
5. Add-student.html – Used to add students into the student management system
6. Update-Student-academics.html – Used to update student academics

### 4.4.2 CSS STRUCTURE

The CSS styling was crafted to create a modern, and responsive design. The styling includes:

* **Global Styles:** Base styles applied across the entire system, including font settings, color schemes, and layout controls.
* **Responsive Design:** Media queries ensure the system is accessible and functional across various devices, from desktops to smartphones.
* **Component-Specific Styles:** Custom styles for individual components such as navigation bars, cards, forms, and buttons.

Forms and modals were designed to be both functional and visually appealing, ensuring a positive user experience. The design includes:

* **Input Fields:** Styled with consistent padding, borders, and fonts to enhance usability.
* **Buttons:** Designed to be easily identifiable, with clear labels and hover effects.
* **Modals:** Centered, with a semi-transparent background overlay to focus user attention on the modal content.

Media queries were used to adjust the layout, font sizes, and spacing for optimal viewing on both desktop and mobile devices.

* **Grid Layouts:** Flexbox and CSS Grid were used to create responsive grid layouts that adjust based on screen size.
* **Typography:** Font sizes were scaled appropriately for different devices, ensuring readability.
* **Navigation:** Mobile-friendly navigation menus, including collapsible or hamburger menus, were implemented.

Subtle animations and effects were added to enhance the user experience without overwhelming the interface. These include:

* **Hover Effects:** Buttons, links, and cards change color or transform slightly when hovered.
* **Transitions:** Smooth transitions were applied to elements such as modals and dropdowns.
* **Loading Animations:** Spinners or progress indicators were used during data loading operations.

## 4.5 JAVASCRIPT IMPLEMENTATION

JavaScript was used throughout the system to enhance user experience by enabling dynamic behaviors and real-time updates. Key features implemented using JavaScript include:

* **Form Validations:** Ensuring user inputs meet required criteria before submission.
* **Dynamic Content Updates:** Real-time updates to the UI without requiring a full page reload.
* **AJAX Requests:** Communication with the server to retrieve or send data asynchronously.
* **Interactive Components:** Enhancing UI components like modals, dropdowns, and tooltips.

To ensure the system remains fast and responsive, JavaScript code was optimized for performance. Techniques like code minification, lazy loading, and event delegation were employed.

* **Code Minification:** JavaScript files were minified to reduce their size and improve load times.
* **Lazy Loading:** Non-critical resources were loaded only when needed, improving initial page load performance.
* **Event Delegation:** Event listeners were managed efficiently to reduce memory usage and improve responsiveness.

# CHAPTER 5: TESTING AND EVALUATION

## 5.1TESTING STRATEGY

The goal of the Student Management System testing technique was to make sure that the system meets all requirements, functions dependably, and offers a smooth user experience. The testing strategy, procedures, and instruments employed during the development process are described in this section, structure and manage database interactions.

### TESTING OBJECTIVES

The primary objectives of the testing strategy are to:

* **Ensure Functional Correctness:** Verify that all features and functionalities work as intended.
* **Identify and Resolve Defects:** Detect and fix any bugs or issues before deployment.
* **Validate User Experience:** Ensure the system is user-friendly, with a consistent and responsive interface.
* **Confirm Performance and Security:** Test the system's performance under various conditions and ensure it is secure from vulnerabilities.

### TESTING METHODOLOGIES

The following testing methodologies were employed to achieve the testing objectives:

* **Unit Testing:** Individual components and functions were tested in isolation to ensure they perform correctly. This was primarily conducted using automated tests.
* **Integration Testing:** Multiple components were tested together to ensure they interact correctly. This included testing the interaction between the front-end and back-end systems.
* **System Testing:** The entire system was tested as a whole to verify that it meets the specified requirements and functions as expected.
* **User Acceptance Testing (UAT):** End-users tested the system in a real-world scenario to ensure it meets their needs and expectations.
* **Performance Testing:** The system was tested under various loads to ensure it performs efficiently. This included stress testing, load testing, and scalability testing.
* **Security Testing:** The system was evaluated for potential security vulnerabilities, including SQL injection, XSS attacks, and data encryption weaknesses.

### 5.1.3 TESTING PROCESS

The testing process was iterative and closely integrated with the development lifecycle. The key steps included:

* **Test Planning:** Identifying the scope, objectives, and resources required for testing.
* **Test Design:** Creating detailed test cases and scenarios based on system requirements.
* **Test Execution:** Running the test cases and recording the results. Automated tests were executed regularly to catch regressions.
* **Defect Tracking:** Any issues or bugs discovered during testing were logged in a defect tracking system (e.g., Jira) and assigned for resolution.
* **Regression Testing:** After any changes or bug fixes, regression tests were conducted to ensure that new issues were not introduced.
* **Test Reporting:** Test results were documented, and regular reports were generated to track progress and coverage.

### 5.1.4 TEST COVERAGE

The testing strategy aimed to achieve comprehensive test coverage, including:

* **Functional Coverage:** All features and functionalities were covered by test cases.
* **Code Coverage:** Automated tests were designed to cover a significant percentage of the codebase.
* **User Scenario Coverage:** Common user workflows were thoroughly tested, including edge cases and error conditions.

### 5.1.5 TEST ENVIRONMENT

Testing was conducted in an environment that closely mirrored the production environment. The test environment included:

* **Servers:** Configured to replicate the production setup, including web servers, database servers, and application servers.
* **Test Data:** A representative dataset was used to simulate real-world usage.
* **Browser and Device Testing:** The system was tested across different browsers (e.g., Chrome, Firefox, Edge) and devices (e.g., desktops, tablets, smartphones) to ensure compatibility.

## TEST CASES

This section outlines the test cases used to validate the various functionalities of the Student Management System. Each test case is designed to ensure that a specific feature or aspect of the system behaves as expected under defined conditions.

### TEST CASE STRUCTURE

Each test case includes the following details:

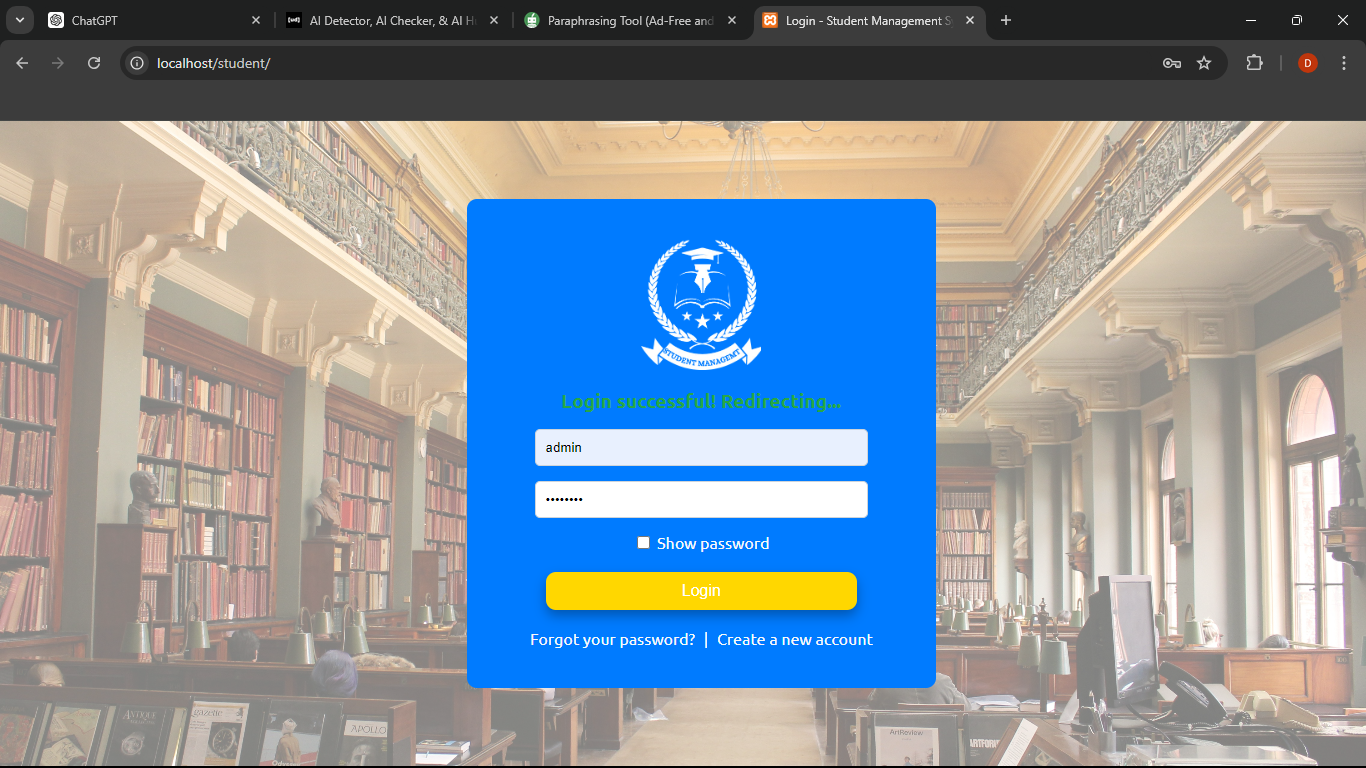
* **Test Case ID:** A unique identifier for the test case.
* **Test Description:** A brief description of what the test case is intended to validate.
* **Preconditions:** Any setup or prerequisites required before executing the test.
* **Test Steps:** Detailed instructions on how to perform the test.
* **Expected Result:** The anticipated outcome if the system behaves correctly.
* **Actual Result:** The observed outcome during testing.
* **Pass/Fail Status:** An indication of whether the test passed or failed based on the comparison of expected and actual results.

### SAMPLE TEST CASES

Test Case 1  
**Test Description:** Verify successful login with valid credentials.  
**Preconditions:** The user must be registered and have valid login credentials.  
**Test Steps:**

1. Navigate to the login page.
2. Enter a valid username in the "Username" field.
3. Enter a valid password in the "Password" field.
4. Click the "Login" button.

**Expected Result:** The user is redirected to the dashboard page.

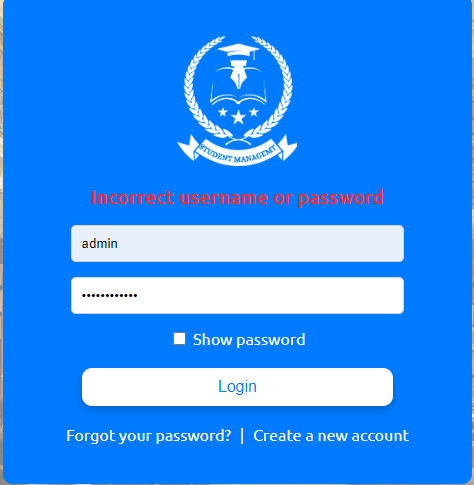


#### Test Case 2

**Test Description:** Verify login failure with invalid credentials.  
**Preconditions:** The user must attempt to log in with incorrect credentials.  
**Test Steps:**

1. Navigate to the login page.
2. Enter an invalid username or password.
3. Click the "Login" button.

**Expected Result:** The system displays an error message indicating invalid credentials, and the user remains on the login page.



## 5.3 EVALUATION

The evaluation of the Student Management System's testing phase indicates that the system largely meets its functional, performance, and security requirements. Most test cases were successfully executed, demonstrating that the system operates as intended under normal conditions.

**Key Findings:**

* **Functionality:** The system’s core features, such as student registration, attendance tracking, and report generation, function correctly. Minor issues were identified but were quickly resolved.
* **Performance:** The system performs well under typical usage but shows some slowdowns under heavy load, which were addressed with optimizations.
* **Security:** Security testing revealed a few vulnerabilities, which were promptly fixed to ensure data safety.
* **User Experience:** The feedback from User Acceptance Testing (UAT) led to improvements in the interface, making it more user-friendly.

**Conclusion:**

Overall, the system is robust, secure, and user-friendly. While a few issues were identified during testing, they were addressed efficiently. The system is ready for deployment, with ongoing monitoring recommended to maintain performance and security.

# CHAPTER 6: DEPLOYMENT

## DEPLOYMENT PROCESS

The deployment process for the Student Management System involves a series of steps to ensure a smooth transition from development to a live production environment. Below are the key steps:

#### 1. **Preparation**

* **Code Review:** Ensure all code has been reviewed, tested, and approved.
* **Backup:** Create backups of the current production environment, including databases and files.
* **Environment Setup:** Prepare the production server by configuring the necessary software (e.g., PHP, MySQL, web server).

#### 2. **Deployment**

* **Transfer Files:** Upload the application files (HTML, CSS, JavaScript, PHP) from the development environment to the production server.
* **Database Migration:** Import the latest database structure and data into the production database.
* **Configuration:** Update configuration files (e.g., database connection settings) to match the production environment.
* **Run Migrations:** Execute any database migration scripts to ensure the database schema is up to date.

#### 3. **Testing**

* **Smoke Testing:** Perform basic tests to ensure the application is running correctly after deployment (e.g., login, navigation, form submissions).
* **Security Checks:** Verify that security measures (e.g., SSL, firewall settings) are properly configured.
* **Performance Testing:** Check that the application performs well under typical usage conditions.

#### 4. **Launch**

* **Go Live:** Once testing is successful, switch the system to live mode, making it accessible to users.
* **Monitoring:** Monitor the system closely for any issues immediately after launch.

#### 5. **Post-Deployment**

* **User Training:** Provide training sessions for end-users to familiarize them with the new system.
* **Documentation:** Ensure all deployment steps are documented for future reference.
* **Ongoing Support:** Offer ongoing technical support to address any issues that may arise post-deployment.

## 6.2 MAINTENANCE

The maintenance phase is essential for ensuring the continued reliability, security, and performance of the Student Management System. It involves regular monitoring, updates, and support to address any issues that arise after deployment.

#### 1. **Regular Monitoring**

* **System Health Checks:** Continuously monitor the system's performance, server load, and database activity to detect and resolve potential issues early.
* **Error Logs:** Review application and server error logs regularly to identify and fix errors or warnings.

#### 2. **Security Updates**

* **Patch Management:** Apply security patches and updates to the system, server, and software dependencies (e.g., PHP, MySQL) to protect against vulnerabilities.
* **User Access Reviews:** Periodically review user accounts and permissions to ensure only authorized personnel have access to the system.

#### 3. **Backup and Recovery**

* **Regular Backups:** Schedule regular backups of the database and application files to prevent data loss. Store backups securely, both on-site and off-site.
* **Disaster Recovery:** Test the disaster recovery plan periodically to ensure that the system can be restored quickly in the event of a failure.

#### 4. **Performance Optimization**

* **Database Maintenance:** Perform routine database maintenance, such as indexing and cleaning up old records, to optimize performance.
* **Code Optimization:** Regularly review and optimize the codebase to improve system efficiency and reduce load times.

#### 5. **User Support**

* **Help Desk:** Provide ongoing user support through a help desk or ticketing system to address any user-reported issues or questions.
* **Training:** Offer refresher training sessions for users to keep them updated on new features and best practices.

#### 6. **System Updates**

* **Feature Enhancements:** Implement new features or enhancements based on user feedback and evolving requirements.
* **Version Upgrades:** Plan and execute system upgrades to newer versions of the software platform to benefit from new capabilities and improvements.

#### 7. **Documentation**

* **Update Documentation:** Keep all system documentation up to date, including user manuals, technical guides, and maintenance logs.

# CHAPTER 7: CONCLUSION AND FUTURE WORK

## 7.1 SUMMARY

The Student Management System project was initiated to enhance the efficiency, accuracy, and accessibility of student data management within the educational institution. Throughout the development process, the project successfully met its objectives by delivering a robust, secure, and user-friendly system.

Key accomplishments include:

* **System Development:** The system was developed using modern technologies, following best practices in coding standards, security, and user experience design.
* **Core Features:** Essential features, such as student registration, attendance tracking, notifications, and report generation, were implemented and thoroughly tested.
* **Deployment:** The system was successfully deployed to a live environment, with a comprehensive deployment process ensuring a smooth transition.
* **Maintenance:** Ongoing maintenance plans were established to keep the system secure, up-to-date, and performing optimally.

This project has a solid foundation for the efficient management of student data, contributing to improved administrative processes and better access to information for staff and students alike.

## 7.2 FUTURE ENHANCEMENTS

While the Student Management System meets the current requirements, there are several opportunities for future enhancements to further improve its functionality, user experience, and scalability. Potential future enhancements include:

#### 1. **Mobile Application**

* Develop a mobile application version of the system to allow students, teachers, and administrators to access and manage information on the go.

#### 2. **Advanced Analytics**

* Implement advanced analytics features, such as predictive attendance patterns and academic performance trends, to support data-driven decision-making.

#### 3. **Parent/Guardian Portal**

* Introduce a dedicated portal for parents or guardians to monitor their child’s attendance, academic progress, and receive important notifications.

#### 4. **Integration with Learning Management Systems (LMS)**

* Integrate the system with popular Learning Management Systems (LMS) to streamline the management of academic content and student performance tracking.

#### 5. **Automated Reporting**

* Enhance the reporting module to allow for automated generation and distribution of reports to relevant stakeholders at scheduled intervals.

#### 6. **Cloud-Based Deployment**

* Explore the possibility of migrating the system to a cloud-based environment to improve scalability, accessibility, and disaster recovery capabilities.

#### 7. **AI-Powered Chatbot**

* Implement an AI-powered chatbot to assist users with common queries, provide guidance on using the system, and improve overall user support.

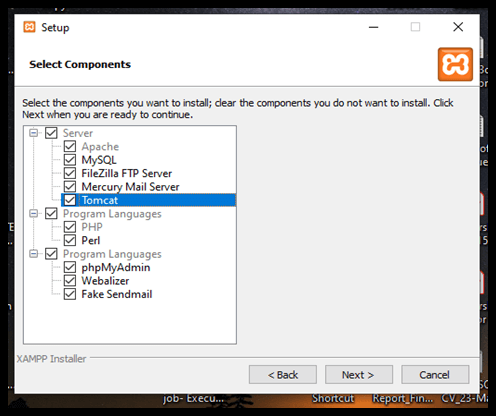
#### 8. **Customizable User Roles**

* Develop more granular and customizable user roles and permissions to better accommodate the diverse needs of different user groups within the institution.

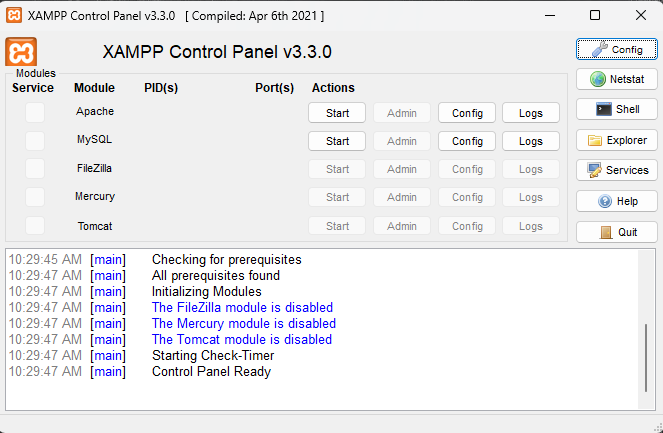
# USER-MANUAL

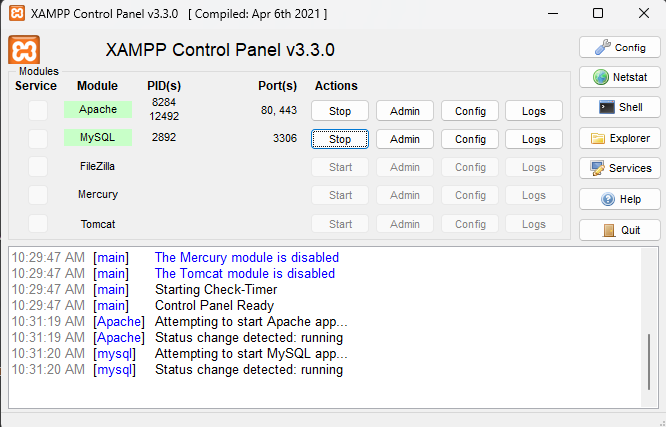
#### **1. Installing XAMPP**

1. **Download XAMPP:**
   * Visit the XAMPP website and download the version suitable for your operating system.
2. **Install XAMPP:**
   * Run the installer and follow the on-screen instructions.
   * Choose the default components (Apache, MySQL, PHP) and installation directory (usually C:\xampp).

****

1. **Start XAMPP:**
   * Open the XAMPP Control Panel from your installation directory or Start menu.
   * Click the "Start" buttons next to "Apache" and "MySQL" to start the web server and database.



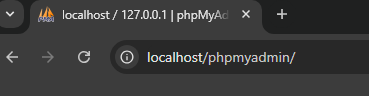


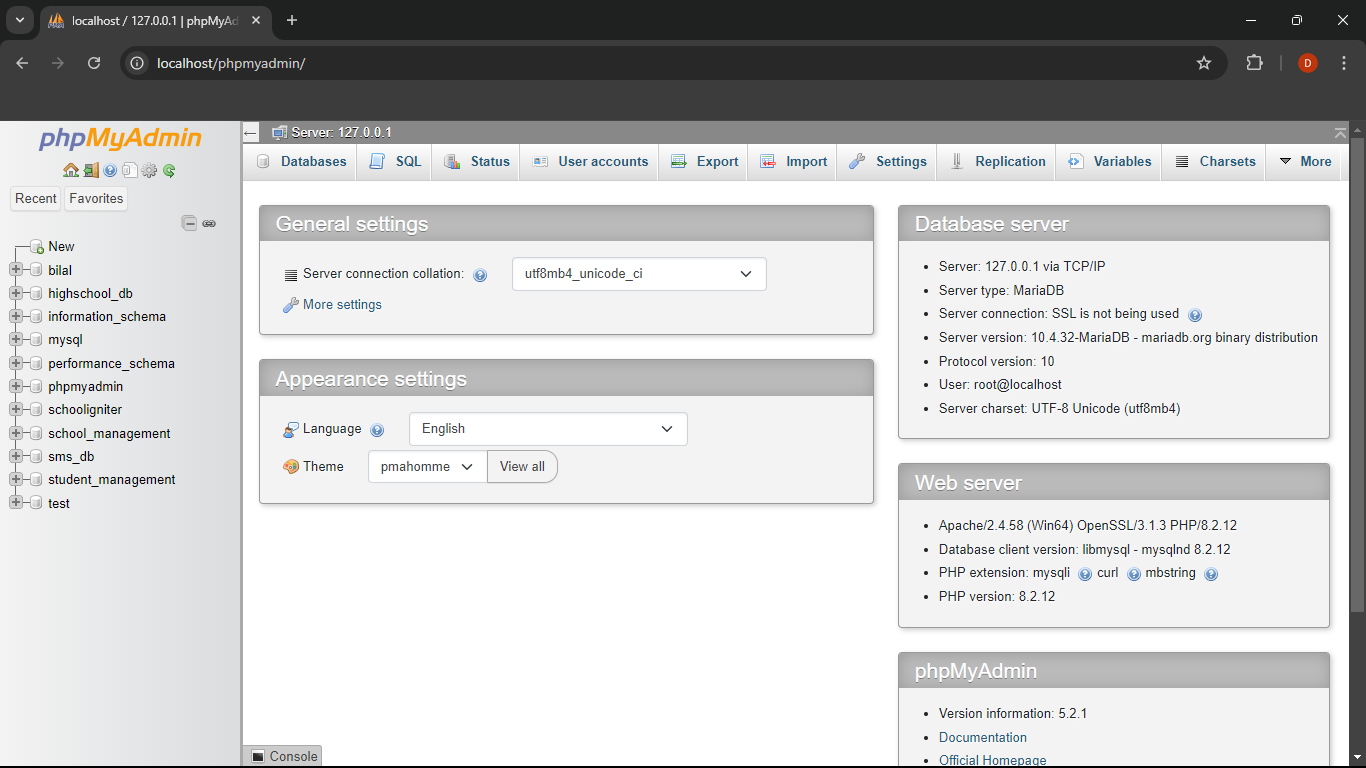
#### **2. Setting Up the Student Management System**

1. **Extract the student.zip File:**
   * Locate the student.zip file you received.
   * Right-click on the file and select "Extract All" to extract the contents.
   * Choose a destination folder for the extracted files.
2. **Move the Extracted Folder:**
   * After extraction, you will have a folder named student.
   * Cut the student folder and navigate to the XAMPP installation directory (usually C:\xampp\).
   * Open the htdocs folder inside the XAMPP directory and paste the student folder here.

#### **3. Configuring the Database Connection**

1. **Access phpMyAdmin:**
   * Open your web browser and type http://localhost/phpmyadmin in the address bar to access phpMyAdmin.

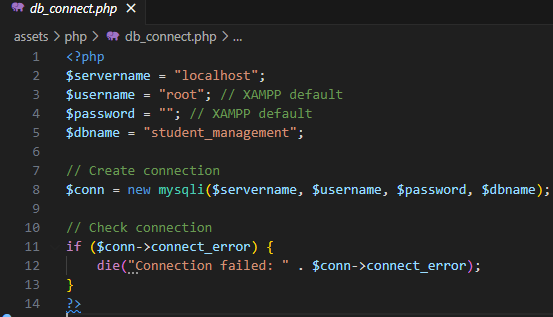




1. **Copy Server Name or Address:**
   * In phpMyAdmin, look for the server information, which is usually displayed on the right side of the homepage.
   * Copy the server name or address (e.g., localhost).

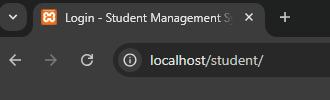


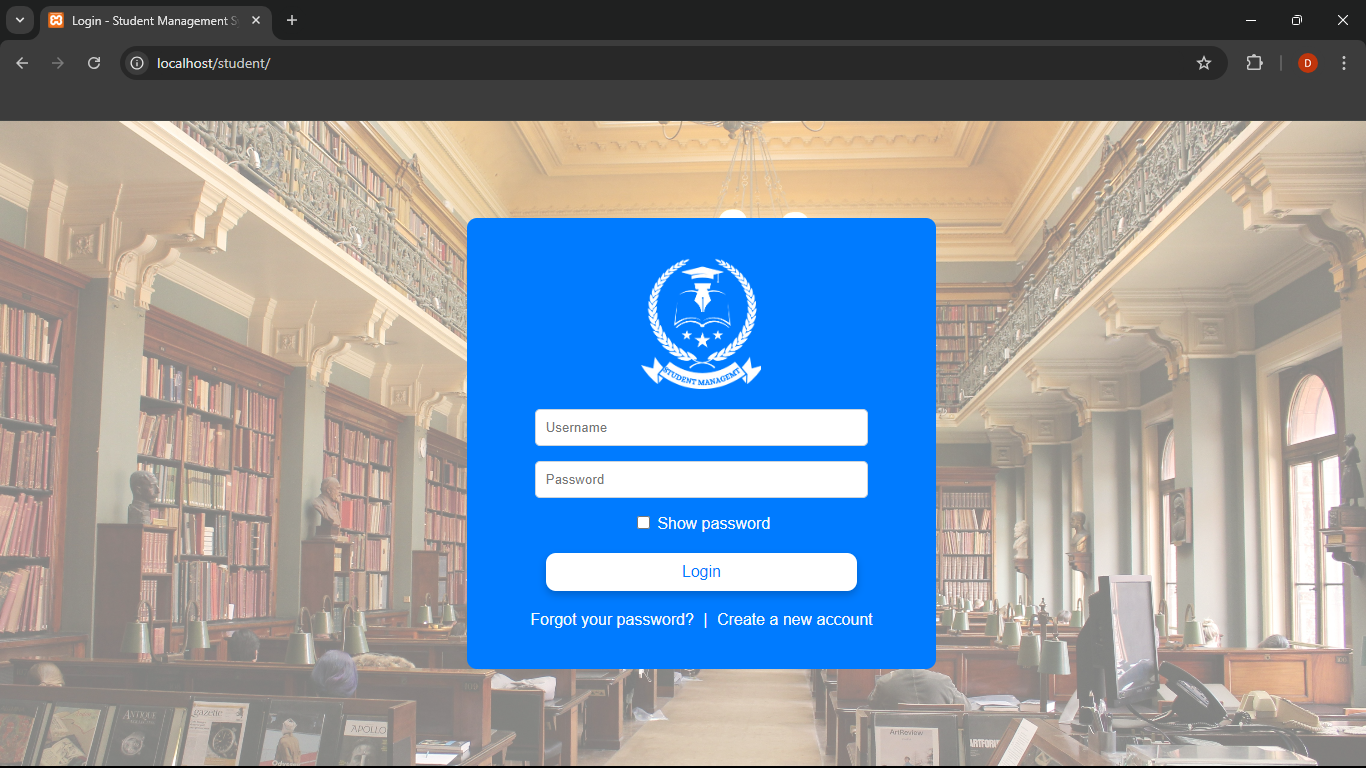
1. **Edit db\_connect.php:**
   * Navigate to the student folder inside htdocs and find the assets/php/db\_connect.php file.
   * Open db\_connect.php in a text editor.
   * Paste the copied server name or address into the $servername variable.



#### **4. Setting Up the Database**

1. **Create a New Database:**
   * In phpMyAdmin, click on the "Databases" tab at the top.
   * Enter student\_management as the database name and click "Create."
2. **Import the Database File:**
   * With the student\_management database selected, click on the "Import" tab.
   * Click "Choose File" and select the database.sql file from the database folder inside the student folder.
   * Click "Go" to import the SQL file and create the necessary tables.**5. Running the Website**
3. **Access the Website:**
   * Open your web browser and type http://localhost/student in the address bar.
   * The Student Management System should now be accessible





# REFERENCES

1. **PHP Documentation**  
   PHP: Hypertext Preprocessor. Retrieved from <https://www.php.net/docs.php>
2. **MySQL Documentation**  
   MySQL 8.0 Reference Manual. Retrieved from <https://dev.mysql.com/doc/refman/8.0/en/>
3. **HTML & CSS Standards**  
   W3C HTML5 Specification. Retrieved from <https://www.w3.org/TR/html5/>  
   W3C CSS Specification. Retrieved from <https://www.w3.org/Style/CSS/Overview.en.html>
4. **JavaScript Documentation**  
   Mozilla Developer Network (MDN) Web Docs. JavaScript. Retrieved from <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
5. **Bootstrap Framework**  
   Bootstrap Documentation. Retrieved from https://getbootstrap.com/docs/
6. **Font Awesome Icons**  
   Font Awesome Documentation. Retrieved from https://fontawesome.com/docs
7. **GitHub**  
   GitHub Repository for Version Control. Retrieved from <https://github.com/>
8. **User Feedback and Testing Reports**  
   Internal testing documents and user feedback collected during the User Acceptance Testing (UAT) phase.
9. **Security Best Practices**  
   OWASP Top Ten Security Risks. Retrieved from https://owasp.org/www-project-top-ten/

# APPENDICES