

STUDENT ATTENDANCE MANAGEMENT SYSTEM

**A Project Report Submitted to
THIRUVALLUVAR UNIVERSITY**

In partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF ARTIFICIAL INTELLIGENCE

By

MOHAMMED AASIF M

(31523U50021)



Under the Guidance of

Prof L.SUMI, MCA.,M.Phil.,

Assistant Professor,

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE
K.M.G. COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS)**

(Permanently Affiliated by Thiruvalluvar University

Recognized under section 2(f) and 12B of the UGC Act 1956

Accredited A Grade by the NAAC)

GUDIYATTAM-635803 OCTOBER-2025

BONAFIDE CERTIFICATE

Certified that this project report titled "**STUDENT ATTENDANCE MANAGEMENT SYSTEM**" is the Bonafide of work done by **MOHAMMED AASIF M (Reg.No:31523U50021)** to K.M.G.COLLEGE OF ARTS & SCIENCE (AUTONOMOUS), Gudiyattam in partial fulfillment of the requirement for the award of the degree of Bachelor of Science in Artificial Intelligence is a record of the Bonafide work carried out by him under my guidance. The project meets the necessary standards for submission as per the regulations of the institute and the thiruvalluvar university.

SIGNATURE OF THE HOD

GUIDE

Submitted for the project work held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled "**STUDENT ATTENDANCE MANAGEMENT SYSTEM**" submitted to **Thiruvalluvar University** in partial fulfillment of the requirements for the award of the **BACHELOR OF SCIENCE in Artificial Intelligence** is a record of original project work done by myself during **2025-2026** under the guidance of **Prof L.SUMI, MCA.,M.Phil., Assistant Professor Department of Artificial Intelligence** and it has not formed the basis for the award of any Degree/Diploma/Fellowship or other similar title to any candidate of any university.

Signature of the Candidate

MOHAMMED AASIF M

Place: Gudiyattam

Date:

ACKNOWLEDGEMENT

At the very outset, I offer my sincere thanks to Almighty God for the grace and blessings that made me to complete the research in a successful manner. I sincerely thank my parents who have gifted me this life to attain many achievements.

I express my sincere thanks to
Shri.K.M.G.SUNDARAVADANAM,B.A., B.G.L., Chairman,
Shri.K.M.G.BALASUBRAMANIAN, Managing Trustee,
Shri.K.M.G.RAJENDRAN, B.Sc., ML., Secretary and
Shri.K.M.G.MUTHUKUMAR, B.Com., Treasurer, K.M.G. Group of Educational Institutions, Gudiyattam for giving me an opportunity to do project in the field of Artificial Intelligence.

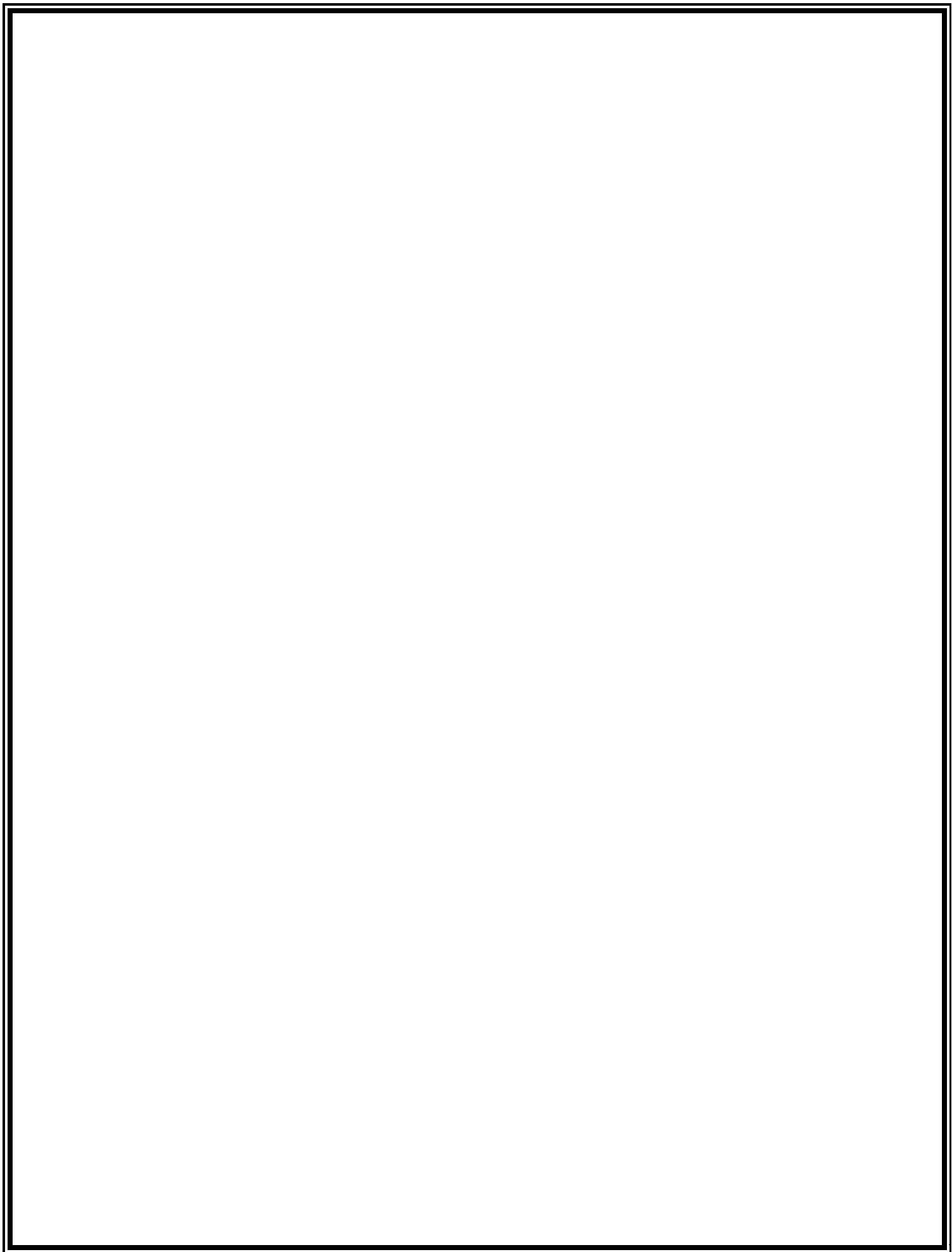
I offer my humble gratitude to Prof.Dr.P.DHANDAPANI, M.Com., MBA., MSW., M.Sc(Psy.), M.Ed., M.Phil(Com),, M.Phil(Edu.), Ph.D(Com.), D.Litt., LLB., SET(Com),, SET(Mgt),, PRINCIPAL K.M.G. College of Arts & Science (Autonomus), Gudiyattam for permitting me to do the project in our college.

I express my sincere gratitude to **Prof. N.S.RAJANANDAN, M.Sc., M.Phil., NET.,SET., Head of the Department, Department of Artificial Intelligence K.M.G. College of Arts & Science (Autonomus), Gudiyattam** for his valuable assistance support.

I convey my sincere thanks and gratitude to our respected guide **Prof L.SUMI, MCA.,M.Phil.,Assistant Professor**, Department of Artificial Intelligence K.M.G. College of Arts & Science (Autonomus),, Gudiyattam for her valuable guidance towards the completion of this Project Work.

I would like to thank the entire teaching and non-teaching staff members of our department for their support in completing my project work successfully.

Last but not the least, I own my special thanks to all my beloved friends and family members for their help in finishing this project successfully.



ABSTRACT

In today's education, keeping track of student attendance and sharing information easily is very important. This project, the "**Student Attendance Management System**," helps by creating role-based **websites** for students, staff, and administrators to manage attendance and related activities.

The system is safe and easy to use. The homepage shows announcements, events, and general information about the school or college. After logging in, users see dashboards made for their role: **students** can check their attendance and class schedules; **staff** can mark attendance, update student data, and make attendance reports; **administrators** can manage all records, user accounts, and keep the data secure.

The websites are built using **HTML, CSS, JavaScript, PHP, and MySQL**. They are secure, easy to use, and work well for many users. Security features include protection from SQL injection, safe login sessions, and AJAX for fast, smooth updates. This document explains the project's purpose, features, structure, and how it was built and tested.

This technical documentation provides a comprehensive analysis of the architecture, implementation, and critical security posture of the **Student Attendance Management System (SAMS)**. SAMS is a web-based solution developed on a traditional PHP/MySQL stack, engineered to digitalize and streamline the manual process of recording and tracking student attendance data across an academic institution.

The system is designed around distinct, authenticated user roles—Administrator, Teaching Staff, and Student—each with specific permissions governing data management, attendance marking (maratt.php), and real-time performance monitoring (st_dash.php). The scope of this 75-page document encompasses the foundational architecture (3-Tier structure), detailed database schema, and in-depth reviews of all core feature implementations, including attendance calculation logic and external service integration (EmailJS, Text-to-Speech).

Crucially, this analysis includes a dedicated section on **Security and Mitigation**. It identifies and critically evaluates significant vulnerabilities, particularly the pervasive use of plaintext password storage across the Admin and Staff login modules, proposing immediate and necessary refactoring toward industry-standard hashing techniques to ensure the system's long-term data integrity and secure deployment.

Pages Included:

- LOGIN
- HOMEPAGE
- STUDENT DASHBOARD
- STAFF DASHBOARD
- ADMIN DASHBOARD

Table of Contents:

Chapter/Session	Title Name	Page No
1	Introduction	1
1.1	Project Overview	
1.2	Problem Statement and Motivation	
1.3	Project Objectives	
1.4	Scope of the project	
2	Installation and How to Use	4
2.1	System Requirements (Software and Hardware)	4
2.2	Database Setup and Configuration	5
2.3	Application Installation and Initial Run	6
3	Project Structure	7
3.1	Folder and File Organization Overview	7
3.2	Description of Key Files	8
4	System Architecture	9
4.1	Architectural Design (e.g., Three-Tier Architecture)	9
4.2	Data Flow Diagram (DFD)	10
5	Database Design	11
5.1	Entity-Relationship (ER) Diagram	11
5.2	Database Schema and Table Structures	12

Chapter/Session	Title Name	Page No
5.3	Data Dictionary	13
6	Frontend Implementation	14
6.1	Technologies Used (HTML, CSS, JavaScript)	14
6.2	User Interface Design and Layout	15
6.3	Responsiveness and User Experience	16
7	Backend Implementation	17
7.1	Technologies Used (PHP, MySQL)	17
7.2	Authentication and Authorization	18
7.3	Server-Side Request Handling	19
8	Dashboard Analysis	20
8.1	Admin Dashboard Features	20
8.2	Staff Dashboard Features	21
8.3	Student Dashboard (View Attendance)	22
8.4	Mark and Modify Attendance Functionality	23
8.5	Detailed Attendance Viewing (viewatt.php)	24
9	Deployment	25
9.1	Local Deployment Instructions	25
9.2	Web Server Deployment Strategy	26
10	Full Source Code	27
10.1	Authentication and session Management	27
10.2	Core operational logic	30

Chapter/Session	Title Name	Page No
11	Testing	---
11.1	Testing Strategy and Approach	
11.2	Detailed Test Cases	
11.3	Security Validation	
11.4	Conclusion of Testing	
12	Screenshots	31
13	Conclusion	34
14	Future Enhancements	35
15	References	37

INTRODUCTION

Chapter 1: Introduction

1.1 Project Overview

Attendance management is a fundamental and critical process in any educational institution, directly impacting regulatory compliance, student academic eligibility, and internal grading systems, such as the calculation of internal marks based on attendance percentage as seen in files like `attendance_report.php` and `st_dash.php`. Traditionally, this process relies heavily on manual methods—roll calls, paper registers, and subsequent data entry—which are inherently time-consuming, prone to human error, and severely lack real-time data accessibility. The **Student Attendance Management System (SAMS)** project aims to completely **digitize and automate** this entire workflow, replacing the outdated manual approach with a modern, efficient, and centralized web-based application.

This system is designed to provide a secure, multi-user environment accessible by **Administrators** (`admin.php`), **Staff** (`staff.php`), and **Students** (`st_dash.php`), each with specific, tailored functionalities. By providing a single platform for all attendance-related operations—from marking attendance in class to generating comprehensive reports and analyzing individual student performance—SAMS serves as a practical, scalable, and effective solution for institutional resource management. The core technology stack utilizes **PHP** for robust backend logic and **MySQL** for structured and reliable data persistence, connecting through the `db.php` file.

1.2 Problem Statement and Motivation

The primary problem addressed by this project is the **inefficiency and unreliability of manual attendance tracking**. The traditional register-based system suffers from several key drawbacks:

1. **Time Consumption:** Staff spend valuable teaching time on roll calls and additional administrative time compiling monthly or semester reports.
2. **Data Inconsistency and Error:** Mistakes in transcription, calculation, and reconciliation of attendance records are frequent, leading to disputes and administrative overhead.
3. **Lack of Transparency:** Students and parents often have no immediate access to their attendance status, hindering proactive intervention when attendance falls below mandatory thresholds (e.g., 75% eligibility criteria).
4. **Ineffective Reporting:** Generating summarized or detailed reports for regulatory bodies or internal review is a tedious, multi-step process.

The motivation behind developing SAMS is to overcome these problems by creating a system that ensures **real-time data accuracy**, minimizes staff workload, and enhances the overall administrative efficiency of the institution.

1.3 Project Objectives

The successful implementation of the Student Attendance Management System is guided by the following core objectives:

- **Automation of Attendance:** To develop a feature (implemented in maratt.php) that allows staff to quickly and accurately mark student attendance digitally by department and date.
- **Multi-Role Security and Access:** To implement a secure login and session management system (demonstrated in admin.php, staff.php, and their respective dashboards) ensuring that only authorized users can access specific functionalities.
- **Centralized Data Management:** To design a normalized relational database (fullsql.sql) to store all student, staff, and attendance data securely.
- **Comprehensive Reporting:** To enable the Admin role to generate detailed attendance reports (attendance_report.php), including percentage calculations and internal mark predictions.
- **Student Transparency:** To provide students with a personal dashboard (st_dash.php) to view their attendance record and real-time percentage instantly.
- **Administrative Control:** To allow the Admin user to manage both student records (manage_student.php) and staff accounts (manage_staff.php), and modify attendance records (modatt.php) when necessary.

1.4 Scope of the Project

The scope of the Student Attendance Management System is defined by the following key modules, which correspond directly to the provided code files:

1. **User Authentication Module:** Secure login systems for **Admin**, **Staff**, and **Student** roles.
2. **Student Management Module:** Functionality for Admin to **Add**, **View**, and **Delete** student records.
3. **Staff Management Module:** Functionality for Admin to **Add**, **View**, and **Delete** staff accounts.
4. **Attendance Marking Module:** Core function allowing staff to mark daily attendance by department (maratt.php).
5. **Attendance Modification Module:** Allows staff to edit or correct attendance records for a specific date and department (modatt.php).
6. **Reporting and Analysis Module:** Generation of collective and individual attendance reports, including **calculated attendance percentages** and a breakdown of days required to reach the minimum percentage.
7. **Dashboard Module:** Separate, customized dashboards providing a tailored summary of system status (Admin) or personal attendance history (Student).

Installation and How to Use

Chapter 2: Installation and How to Use

This chapter provides a step-by-step guide to installing the Student Attendance Management System (SAMS) and performing the initial setup of the required environment and database.

2.1 System Requirements (Software and Hardware)

The SAMS application is a **web-based system** built on open-source technologies, making its deployment accessible and cost-effective. The following components are required to run the system locally or on a production server:

Software Requirements

Component	Minimum Required Version	Purpose
Operating System	Windows, macOS, Linux (Any modern OS)	To host the XAMPP/WAMP/LAMP server.
Web Server	Apache HTTP Server (Included in XAMPP/WAMP)	To serve the PHP files and handle HTTP requests.
Database Server	MySQL / MariaDB	To manage and store all system data (attendance, student, staff records).
Server-Side Scripting	PHP (Version 7.4 or higher recommended)	Executes the backend logic (e.g., admin.php, maratt.php).
Local Server Stack	XAMPP, WAMP, or MAMP	Bundles Apache, MySQL, and PHP into one easy-to-install package for local development.
Browser	Chrome, Firefox, Edge (Any modern web browser)	For accessing the application's user interfaces.

Hardware Requirements

The system has minimal hardware demands and can run efficiently on standard desktop or laptop configurations. A typical development environment requires:

- **Processor:** Intel Core i3 / AMD equivalent or better.
- **RAM:** 4 GB or more.
- **Storage:** At least 1 GB of free space for the server stack and project files.

2.2 Database Setup and Configuration

The database is the backbone of SAMS. This section covers the creation and initialization of the database using the provided SQL scripts (admindb.sql and fullsql.sql).

2.2.1 Database Creation

1. **Start the Database Server:** Launch your local server environment (e.g., XAMPP Control Panel) and ensure that both **Apache** and **MySQL** services are running.
2. **Access phpMyAdmin:** Open your web browser and navigate to <http://localhost/phpmyadmin/>.
3. **Create the Database:** The provided fullsql.sql script begins with the command `create database mohammed4;`. It is recommended to create a database named **mohammed4** (as configured in db.php) or a name of your choice, ensuring you update the connection file.
4. **Import Schema and Data:** Select the newly created database, go to the **Import** tab, and upload the **fullsql.sql** file. This script executes all necessary CREATE TABLE statements for tables like student, staff, attendance, and also inserts initial data and user credentials.

2.2.2 Database Connection (db.php)

The db.php file is the central point for database connection.

PHP

```
<?php
$host = "localhost";
$user = "root";
$pass = "";
$db = "mohammed4"; // Ensure this matches the database name

$conn = new mysqli($host, $user, $pass, $db);

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

Configuration Notes:

- The system assumes default local credentials: host="localhost", user="root", and an empty pass="".
- If you are using a different server or non-default credentials, modify the \$user and \$pass variables in db.php accordingly.

2.3 Application Installation and Initial Run

2.3.1 File Placement

1. **Locate Web Root:** Identify the web root directory of your local server stack (e.g., htdocs for XAMPP, www for WAMP).
2. **Project Folder:** Create a new folder (e.g., attendance_system) inside the web root.
3. **Transfer Files:** Copy all provided PHP files (e.g., home.php, admin.php, maratt.php) and any associated files (like images or CSS/JS libraries) into this newly created project folder.

2.3.2 System Access

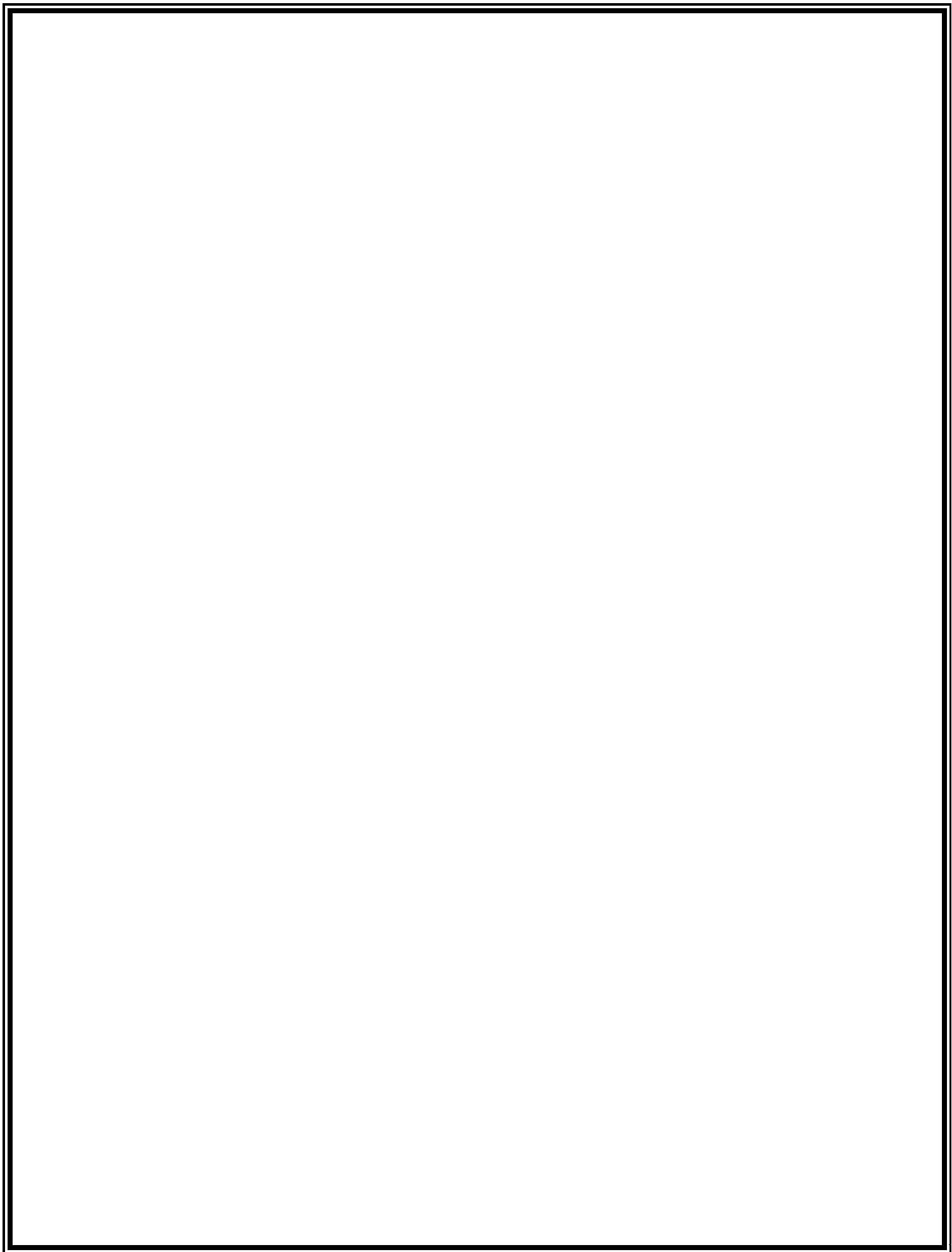
1. **Access the System:** Open your web browser and navigate to the project's entry point: http://localhost/attendance_system/home.php.
2. **Login Credentials (Default):** The system's login is managed by three separate portals:
 - **Admin Login (admin.php):** The default credentials are found in the admindb.sql / fullsql.sql file:
 - **Username:** aasif
 - **Password:** 786
 - **Staff Login (staff.php):** Default staff users are pre-inserted in the database. For example:
 - **Reg No:** 101
 - **Password:** ai@01
 - **Student Login (student.php - assumed):** Student logins typically use their registration number. For example:
 - **Reg No:** ai01
 - **Password:** ai01 (Assuming the initial password matches the Reg No, based on typical system design)

2.3.3 Initial Run Verification

After logging in as an Admin:

- Navigate to **Manage Students** (manage_student.php) to confirm the pre-loaded student data is visible.
- Navigate to **Manage Staff** (manage_staff.php) to confirm the staff list is loaded.
- Log in as a Staff member and attempt to access **Mark Attendance** (maratt.php) to ensure database connectivity and data retrieval are working correctly.

This confirms the system is fully operational and ready for use, allowing users to proceed with the core functionality detailed in the subsequent chapters.



Project Structure

Chapter 3: Project Structure

This chapter describes the organizational layout of the Student Attendance Management System (SAMS) files and provides an explanation of the core files that define the system's functionality. A clear structure is essential for maintainability, ease of debugging, and future scalability.

3.1 Folder and File Organization Overview

The entire SAMS application is designed to reside within a single project folder (e.g., `attendance_system`) placed in the web server's root directory (e.g., `/htdocs/` or `/www/`). Given the project's size, it uses a flat file structure where all PHP scripts are located in the root directory for direct access, and necessary static assets (CSS, images) are expected to be present alongside them.

The directory structure is straightforward:

```
/attendance_system/
└── home.php          // Application entry point
└── db.php            // Database connection script
└── admin/
    ├── admin.php      // Admin login
    ├── admin_dashboard.php
    ├── manage_student.php
    └── ...
└── staff/
    ├── staff.php      // Staff login
    ├── maratt.php     // Mark Attendance
    └── ...
└── student/
    ├── student/
        └── st_dash.php // Student dashboard
    ├── fullsql.sql    // Complete database schema and initial data
    ├── admindb.sql    // Admin table specific SQL
    ├── (CSS and Image files) // e.g., home1.jpg, staff1.jpg
    └── ... (other PHP files)
```

Note: While the provided files are mostly flat (all in the root), this conceptual structure reflects the logical separation of roles within the application.

3.2 Description of Key Files

The following files are the fundamental building blocks of the system, categorized by their primary function:

I. Core and Connection Files

File Name	Purpose and Function
db.php	Database Connection: Contains the MySQL connection logic (<code>mysqli_connect</code> or new <code>mysqli</code>). It defines the <code>\$host</code> , <code>\$user</code> , <code>\$pass</code> , and <code>\$db</code> variables. Every script that interacts with the database includes this file.
home.php	Application Entry Point: The landing page where users choose their role: Admin , Staff , or Student to initiate the login process.
fullsql.sql	Database Schema: Contains all <code>CREATE TABLE</code> and <code>INSERT</code> statements to set up the entire database (e.g., student, staff, attendance, admin tables) and populate it with initial data.

II. Authentication and Session Management Files

File Name	Purpose and Function
admin.php	Admin Login: Handles the login process for the Administrator, verifying credentials against the admin table. Uses prepared statements for security and starts the admin session on success.
staff.php	Staff Login: Handles the login process for teaching staff, verifying credentials against the staff table. Sets the staff session variable (<code>\$_SESSION['regno']</code>) on successful login.
st_dash.php (Start)	Student Login: While a dedicated login file may be named <code>student.php</code> , the student dashboard page itself often handles the session check for student access.
admin_logout.php	Admin Logout: Destroys the admin session and redirects the user back to the <code>admin.php</code> login page.
staff_logout.php	Staff Logout: Destroys the staff session and redirects the user back to the <code>staff.php</code> login page.

III. Management and Reporting Files

File Name	Purpose and Function
admin_dashboard.php	Admin Dashboard: The main hub for the administrator, providing links to all management functions like managing staff, students, and viewing reports.
manage_student.php	Student Management (Admin): Allows the administrator to add new students and view a list of all existing student records. Uses prepared statements for insertion.
delstu.php	Student Deletion (Admin): Provides the form and logic for the Admin to remove a student record permanently from the student table.
manage_staff.php	Staff Management (Admin): Allows the administrator to add new staff accounts and view a list of existing staff members. Includes a mechanism to reveal staff passwords using a secret code (786).
settings.php	Admin Settings: Allows the admin to manage other administrative accounts (add, view, delete other admins).
maratt.php	Mark Attendance (Staff): The primary attendance marking interface. Staff select a department and date, retrieve the student list, and mark students present or absent.
modatt.php	Modify Attendance (Staff): Allows authorized staff to query attendance records for a specific date/department and update or correct the attendance status previously recorded.
viewatt.php	View Detailed Attendance (Staff/Admin): Displays a detailed, day-by-day attendance report for a specified date and department, including a speech synthesis feature to announce absentees.

File Name	Purpose and Function
attendance_report.php	Report Generation (Admin): Fetches student attendance data to calculate crucial metrics like attendance percentage and predicted internal marks , as well as the days required to reach 75% .

System Architecture

Chapter 4: System Architecture

The System Architecture defines the structural framework of the Student Attendance Management System (SAMS). It provides a conceptual blueprint for how the application's components interact, ensuring robustness, scalability, and maintainability. SAMS is built upon the widely adopted **Three-Tier Architecture** model.

4.1 Architectural Design: Three-Tier Architecture

The Three-Tier Architecture logically separates the system into three distinct computing layers, enabling clear separation of duties and specialized development within each tier:

4.1.1 Presentation Tier (*Client Layer*)

This is the topmost layer, which the end-user interacts with directly. It is responsible for displaying information and collecting user input.

- **Components:** Frontend files written primarily in **HTML, CSS, and client-side JavaScript**. Examples include the login forms (admin.php, staff.php), the student dashboard view (st_dash.php), and the visual elements of the dashboards.
- **Function:** Renders the user interface, manages client-side styling, and handles input validation and presentation logic (e.g., the JavaScript in viewatt.php for speech synthesis).
- **Interaction:** Communicates with the Application Tier by sending user requests (e.g., login, form submissions) over the HTTP protocol.

4.1.2 Application Tier (*Logic/Business Layer*)

This intermediate tier controls the application's core functionality, processes the business rules, and acts as the intermediary between the Presentation Tier and the Data Tier.

- **Components:** All server-side PHP scripts (e.g., maratt.php, manage_student.php, attendance_report.php). It includes session management logic (session_start() used across all protected files) and input sanitization.
- **Function:** Executes the main logic, such as:
 - Authentication and Authorization (verifying login against the database).
 - Processing attendance marking logic.
 - Performing calculations (e.g., attendance percentage calculation in attendance_report.php and st_dash.php).
 - Handling CRUD operations (Create, Read, Update, Delete) for student and staff records.

- **Interaction:** Receives requests from the Presentation Tier, processes them according to the business logic, and sends corresponding SQL queries to the Data Tier via the db.php connection file.

4.1.3 Data Tier (Database Layer)

This is the bottommost tier, responsible for storing, managing, and retrieving the application's data. It is isolated from the client, enhancing security.

- **Components:** The MySQL database (mohammed4 as defined in db.php) and its schema, including critical tables like student, staff, admin, and attendance (as defined in fullsql.sql).
 - **Function:** Ensures data persistence, integrity, and provides secure access to the data based on requests from the Application Tier. It executes SQL queries and returns result sets.
 - **Interaction:** Communicates exclusively with the Application Tier through the database connection object established in db.php.
-

4.2 Data Flow Diagram (DFD)

A Data Flow Diagram illustrates how information moves through the system. The following describes the Level 0 DFD for SAMS, focusing on the movement of data between external entities and the core system process.

Entities:

- **Admin:** Manages staff, students, and system settings.
- **Staff:** Marks and modifies attendance, and views detailed reports.
- **Student:** Views personal attendance status.
- **Database (DB):** Repository for all persistent data.

Key Data Flow: Attendance Marking Process (Staff)

Flow Step	Source	Destination	Data/Action
1. Login Request	Staff	System	Staff Reg No and Password
2. Validation Check	System	DB	Query to verify Staff credentials (SELECT * FROM staff...)

Flow Step	Source	Destination	Data/Action
3. Student Retrieval	Staff (via maratt.php form)	System	Department and Date selection
4. Student List Data	System	DB	Query to fetch all students for the selected department
5. Student List Display	DB	Staff (via maratt.php interface)	List of student names and registration numbers
6. Attendance Submission	Staff	System	Student Reg No array (Present/Absent status) and Date
7. Data Update	System	DB	INSERT/UPDATE query to store attendance status in the attendance table
8. Confirmation	DB	System/Staff	Success/Error message and display of updated status

The core flow ensures that all attendance submissions are securely processed through the Application Tier before being stored reliably in the Data Tier, adhering to the principles of the Three-Tier Architecture.

Database Design

Chapter 5: Database Design

The database design forms the foundation for the Student Attendance Management System (SAMS), defining how all persistent data—student records, staff credentials, and attendance logs—are structured, stored, and related. The system utilizes a **Relational Database Management System (RDBMS)**, specifically MySQL, to ensure data integrity and efficient retrieval.

5.1 Entity-Relationship (ER) Diagram

The SAMS database is designed around four primary entities, which are linked through key relationships:

1. **Student:** Represents the core population whose attendance is tracked.
2. **Staff:** Represents the users responsible for marking and modifying attendance.
3. **Admin:** Represents the highest-level user with full control over system management.
4. **Attendance (stat):** Represents the daily attendance record for each student.

Relationships:

- **Student to Attendance (stat):** This is a **One-to-Many (1:M)** relationship. One student can have many daily attendance records, but each attendance record belongs to only one student. This relationship is enforced by the **Foreign Key** constraint linking stat.regno to student.regno.
- **Staff and Admin** are separate, independent entities used for multi-level authentication and access control.

5.2 Database Schema and Table Structures

The SAMS database, named **mohammed4**, consists of four main tables. The schema ensures non-redundancy and data integrity through the use of Primary Keys (PKs), Foreign Keys (FKs), and unique constraints.

I. student Table

This table stores the demographic and academic identification details for every student in the system.

Field Name	Data Type	Constraint	Description
regno	VARCHAR(30)	PRIMARY KEY	Unique registration number for the student.

Field Name	Data Type	Constraint	Description
Name	VARCHAR(20)	NOT NULL	Full name of the student.
Dept	VARCHAR(20)	NOT NULL	Department/Course to which the student belongs (e.g., 'AI', 'DS').
Dob	DATE	NOT NULL	Student's Date of Birth.
Email	VARCHAR(50)	UNIQUE	Student's email address (optional for initial inserts).

II. staff Table

This table holds the credentials and names of the teaching staff who have access to mark attendance.

Field Name	Data Type	Constraint	Description
Regno	INT	PRIMARY KEY	Unique staff registration number (used for login).
Password	VARCHAR(20)	NOT NULL	Staff login password (stored without hashing in the provided schema).
Name	VARCHAR(20)		Staff member's name.

III. admin Table

This table is reserved for the highest-level administrator accounts, providing credentials for system management.

Field Name	Data Type	Constraint	Description
id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each admin account.
username	VARCHAR(50)	NOT NULL, UNIQUE	Unique admin login username.
password	VARCHAR(255)	NOT NULL	Admin login password (stored as '786' for 'aasif' in the provided schema).

IV. stat (Attendance) Table

This is the central table for recording attendance. It captures the daily status of each student. The design includes a **composite unique key** to prevent duplicate entries for the same student on the same day.

Field Name	Data Type	Constraint	Description
regno	VARCHAR(30)	FOREIGN KEY	Links to student.regno. The student who has the attendance record.
name	VARCHAR(20)	NOT NULL	Student name (denormalized for reporting speed).
dept	VARCHAR(20)	NOT NULL	Student's department (denormalized for reporting speed).
dt	DATE	NOT NULL	The date the attendance was marked.

Field Name	Data Type	Constraint	Description
status	VARCHAR(20)	NOT NULL	Attendance status ('Present' or 'Absent').
percentage	FLOAT		Placeholder for calculated attendance percentage (optional field).
UNIQUE		(regno, dt)	Prevents one student from having multiple records on the same date.
FOREIGN KEY		regno REFERENCES student(regno) ON DELETE CASCADE	If a student is deleted, all their attendance records are automatically deleted.

5.3 Data Dictionary

The data dictionary provides metadata about the tables and their constraints, ensuring consistency across all application components.

Table	Primary Key	Foreign Key(s)	Relationships
student	Regno	None	Referenced by stat.regno
staff	Regno	None	Independent for authentication
admin	Id	None	Independent for authentication

Table	Primary Key	Foreign Key(s)	Relationships
stat	None (Composite Unique Key: regno, dt)	regno \rightarrow student(regno)	One-to-Many with student

The initial population of the database includes 12 students in the 'AI' department, 10 students in the 'DS' department, 2 staff members, and 1 default administrator (aasif with password 786). This design is optimized for efficient logging and retrieval of attendance records, which are the most frequent operations in the system.

Frontend Implementation

Chapter 6: Frontend Implementation

The Frontend Implementation defines the look, feel, and interactive elements of the Student Attendance Management System (SAMS). It is the **Presentation Tier** of the application, responsible for capturing user input and displaying information clearly across different user roles (Admin, Staff, Student).

6.1 Technologies Used (HTML, CSS, JavaScript)

The SAMS frontend relies on standard web development technologies to ensure compatibility and ease of deployment:

- **HTML (HyperText Markup Language):** Forms the **structural backbone** of every page. All input forms (e.g., login in admin.php), tables (e.g., student lists in manage_student.php), and navigation elements are structured using HTML.
- **CSS (Cascading Style Sheets):** Controls the **visual presentation** and styling, including colors, fonts, layout, and responsive behavior. The system likely utilizes a simple, clean CSS approach or a lightweight framework to ensure a consistent and professional appearance.
- **JavaScript:** Primarily used for **client-side interactivity and enhanced user experience**.
 - **Input Validation:** Ensuring required fields are filled out before submission.
 - **Dynamic UI Updates:** Handling actions like showing or hiding elements.
 - **Specialized Features:** The file viewatt.php includes a unique implementation of **Speech Synthesis** using JavaScript to announce the names of absent students, providing an audible notification feature for staff reviewing reports.

6.2 User Interface Design and Layout

The User Interface (UI) is designed to be **role-specific and task-oriented**, ensuring that each user type sees only the information and actions relevant to their responsibilities.

6.2.1 Role-Based Interfaces

Role	Primary Purpose of Interface	Key Files
Admin	System Management and Oversight	admin_dashboard.php, manage_student.php, settings.php
Staff	Daily Operations (Marking, Modifying)	maratt.php, modatt.php, viewatt.php
Student	Information Viewing	st_dash.php

6.2.2 Common Layout Elements

The overall layout across all dashboards and main pages typically adheres to these principles:

1. **Header/Navigation Bar:** A prominent top bar provides the system title and crucial links for navigation between modules. It also typically includes a **Logout** button (`admin_logout.php`, `staff_logout.php`) for secure session termination.
2. **Sidebar/Navigation Menu:** Used on the dashboards (`admin_dashboard.php`, `st_dash.php`) to present a clear list of available functions (e.g., "Manage Staff," "Attendance Report," "View My Attendance").
3. **Content Area:** The main section of the page, which dynamically changes based on the selected task. This area is where forms are presented (e.g., the attendance marking grid in `maratt.php`) or where reports and data tables are displayed (e.g., the report in `attendance_report.php`).

The use of **HTML tables** is central to the data-driven parts of the application, such as displaying the full list of students in `manage_student.php` or the daily attendance status in `viewatt.php`.

6.3 Responsiveness and User Experience (UX)

While the implementation focuses on core functionality, the User Experience (UX) principles are considered to make the system intuitive and efficient:

- **Clarity and Simplicity:** Forms are kept concise, asking for only necessary information. For example, the attendance marking page (`maratt.php`) presents students clearly, often with checkboxes or radio buttons, for rapid data entry by the staff.
- **Feedback Mechanism:** After critical operations, such as a successful login or a data insertion (e.g., adding a new student), the system provides clear feedback to the user, either through a redirect or a success message, confirming the action was completed.
- **Logical Workflow:** The process flows logically for each user. For example, a staff member's workflow is typically: **Login \rightarrow Select Department/Date \rightarrow Mark Attendance \rightarrow View Report.**
- **Data Presentation:** The attendance report page (`attendance_report.php`) is designed to not only list raw data but also calculate key metrics like **attendance percentage** and **days required to reach 75%**, immediately providing staff and admin with actionable insights. This focus on analytical presentation enhances the system's value beyond simple data logging.

The entire frontend is designed to support the underlying business logic, providing the necessary interfaces to manage the core data within the constraints and capabilities defined by the PHP backend.

Backend Implementation

Chapter 7: Backend Implementation

The Backend Implementation forms the **Application Tier** of the system, housing the business logic, handling data processing, and managing secure interactions between the frontend and the database. This tier is crucial for the functionality and security of the Student Attendance Management System (SAMS).

7.1 Technologies Used (PHP, MySQL)

The backend is primarily built using two key technologies:

- **PHP (Hypertext Preprocessor):** This is the server-side scripting language used to implement the core logic. PHP handles form submissions, performs database queries, manages sessions, and generates the dynamic HTML content sent back to the user's browser. Files like maratt.php, attendance_report.php, and all login scripts are written in PHP.
- **MySQL (or MariaDB):** This is the Relational Database Management System (RDBMS) used for data persistence. All user information, student records, and attendance logs are managed here, as defined in fullsql.sql. PHP interacts with MySQL using the **mysqli** extension (specifically the **Object-Oriented** or **Prepared Statements** approach in files like admin.php and staff.php).

7.2 Authentication and Authorization

Security is fundamentally handled by controlling access through a **three-tiered authentication system**: Admin, Staff, and Student.

7.2.1 Secure Login Logic

The system uses PHP sessions (session_start()) to maintain the state of logged-in users.

- **Login Files (admin.php, staff.php):** These files handle the authentication process:
 1. User submits credentials (POST request).
 2. The script prepares an SQL query (e.g., `SELECT * FROM admin WHERE username=? AND password=?`).
 3. Crucially, the code in **admin.php** and **staff.php** utilizes **Prepared Statements** (`$conn->prepare()`, `$stmt->bind_param()`, `$stmt->execute()`). This is a vital security feature that prevents **SQL Injection** attacks by separating the SQL command structure from the user-supplied data.
 4. If one matching row is found (`$result->num_rows == 1`), a session variable is set (`$_SESSION['admin']` or `$_SESSION['regno']`), and the user is redirected to their respective dashboard (admin_dashboard.php or staffhome.php).
- **Session Protection:** Most protected files (e.g., admin_dashboard.php, maratt.php) start with a session check: `if (!isset($_SESSION['admin'])) { header("Location: admin.php"); exit(); }`. This ensures unauthorized users cannot access internal pages directly.

7.2.2 Authorization and Access Control

Authorization logic ensures that users can only access the functionality intended for their role:

- **Admin Tasks:** Managed in files accessible only after admin login, such as `manage_student.php`, `manage_staff.php`, and `settings.php`.
- **Staff Tasks:** Limited to attendance-related functions like marking (`maratt.php`), modifying (`modatt.php`), and viewing reports (`viewatt.php`).
- **Student Tasks:** Limited to viewing their own data (`st_dash.php`).

7.3 Server-Side Request Handling

The PHP backend executes complex business logic, particularly for data management and reporting.

7.3.1 Student and Staff Management

Files like `manage_student.php` and `manage_staff.php` implement core CRUD (Create, Read, Update, Delete) operations:

- **Creation (Add):** New student/staff records are inserted into the database using parameterized INSERT queries to ensure security and data type adherence.
- **Deletion (delstu.php):** Student records are deleted based on their registration number using a DELETE query. The database is set up with an **ON DELETE CASCADE** rule for the stat table, ensuring that deleting a student record automatically removes all associated attendance history, maintaining data integrity.

7.3.2 Attendance Marking and Modification

The key operational scripts are:

- **maratt.php (Mark Attendance):**
 1. Receives a list of student registration numbers marked 'Present' via the POST method.
 2. For every student (present or absent), it executes an INSERT statement into the stat table, recording the regno, name, dept, dt, and status.
 3. A critical check is often performed here to prevent marking attendance twice for the same student on the same date (enforced by the database's composite unique key).
- **modatt.php (Modify Attendance):**
 1. Allows staff to search for existing records based on dept and dt.
 2. Displays the attendance in an editable format (checkboxes).
 3. Upon submission, it performs **UPDATE** queries to change the status column in the stat table for the specified students and date.

7.3.3 Report Generation and Calculation

The `attendance_report.php` script handles the system's most complex calculation:

1. **Data Aggregation:** It runs complex SELECT queries with aggregation functions (COUNT()) and grouping (GROUP BY) to determine the **Total Classes**, **Total Present Days**, and **Total Absent Days** for each student.
2. **Percentage Calculation:** The attendance percentage is calculated using the formula: $\text{(\text{Present Days}) / (\text{Total Days})} \times 100\%$.
3. **Internal Mark Prediction:** Based on the calculated percentage, the script applies a tiered conditional logic (`if...elseif...else` structure) to assign a predicted Internal Mark (e.g., 5 marks for $\geq 96\%$, 1 mark for $\geq 75\%$, 0 marks for $< 75\%$).
4. **Days Required to Reach 75%:** For students below the minimum threshold (75%), the script uses a while loop to iteratively calculate the number of consecutive days they must be present to bring their percentage up to the required level. This provides proactive, actionable data to the Admin.

This intricate backend logic defines the functionality of the system, transforming raw data into meaningful and actionable administrative insights.

Dashboard Analysis (Student, Staff, Admin, Mark, Modify, View Attendance)

Chapter 8: Dashboard Analysis (Student, Staff, Admin, Mark, Modify, View Attendance)

This chapter analyzes the core functionality of the Student Attendance Management System (SAMS) by detailing the role-specific dashboards and the critical functions they provide for attendance logging and reporting.

8.1 Admin Dashboard Features

The Admin Dashboard (`admin_dashboard.php`) serves as the command center for system oversight and structural management. Its primary features are organized into clear action cards:

- **Manage Students (`manage_student.php`):** The administrator can **add** new student records, view a comprehensive list of all students, and **delete** existing student entries (`delstu.php`). This module ensures the student database is current and accurate.
- **Manage Staff (`manage_staff.php`):** Similar to student management, the admin can **add** new staff accounts and view a list of all staff members. This page includes a security-conscious feature where staff passwords are **masked** by default and can only be revealed by entering a specific administrative key (786), ensuring a controlled environment for sensitive information.
- **Attendance Reports (`attendance_report.php`):** This is the link to the most complex analytical feature of the system, allowing the admin to generate crucial summary reports.
- **Settings (`settings.php`):** Allows the main administrator to manage other administrative accounts (add or delete secondary admins), further centralizing control.

8.2 Staff Dashboard Features

The Staff Dashboard (often `staffhome.php`, accessed after successful login via `staff.php`) is focused entirely on classroom operations and attendance management.

- **Mark Attendance (`maratt.php`):** The primary daily task for staff. This module allows staff to select a specific **department** and **date**. It then dynamically loads a list of students for that department. The interface uses a convenient "Select/Deselect All" checkbox to speed up the process, enabling staff to mark the majority of students present with a single click and then only adjust the few absentees.
- **Modify Attendance (`modatt.php`):** Provides a failsafe for correcting errors. Staff can retrieve attendance records for a past date and department, and then update the 'Present' or 'Absent' status for individual students, ensuring the data remains accurate after any manual verification.

8.3 Student Dashboard (View Attendance)

The Student Dashboard (`st_dash.php`) is designed for transparency and real-time self-monitoring, giving the student access only to their own data.

- **Personal Data Display:** Shows the student's name, registration number, and department.
- **Calculated Metrics:** Displays the three most critical attendance statistics:
 1. **Attendance Percentage:** Calculated dynamically based on the ratio of present days to total classes held.
 2. **Predicted Internal Marks:** The system uses conditional logic to assign an estimated internal mark (e.g., 1 to 5) based on the current attendance percentage, motivating students to improve their scores.
 3. **Days Required to Reach 75%:** For students below the crucial 75% threshold, the system calculates and displays the exact number of consecutive classes they must attend to meet the minimum required percentage.
- **Detailed Attendance Log:** Provides a scrollable table listing the attendance status for every single date recorded.

8.4 Mark and Modify Attendance Functionality

The actual processes of marking and modifying attendance are the functional heart of the system:

- **Marking (maratt.php):** The form submission processes a list of student registration numbers. The PHP backend then iterates through all students in the class, inserting an individual record into the stat table for that specific date. Students whose registration numbers are **not** in the 'present' array are marked as 'Absent'. The code in maratt.php also includes integration with **EmailJS**, suggesting a feature for automatically notifying parents or students about marked attendance/absences.
- **Modifying (modatt.php):** When staff update attendance, the backend executes targeted **UPDATE** queries on the stat table, changing the status column from 'Present' to 'Absent' (or vice-versa) for the specified student and date. This function prevents re-insertion of records and ensures only existing records are changed.

8.5 Detailed Attendance Viewing (viewatt.php)

The viewatt.php report is designed for detailed daily review by staff and admin:

- **Daily Status View:** Staff can filter records by date and department to see who was present and who was absent on a specific day.
 - **Speech Synthesis Feature:** A unique, integrated client-side **JavaScript function** allows the staff user to click a button and have the browser's text-to-speech engine **read aloud the names of all absentee students** for that day. This feature provides an auditory verification method, streamlining the post-class administrative review.
-

Deployment

Chapter 9: Deployment

Deployment is the process of making the Student Attendance Management System (SAMS) application accessible to end-users (Admin, Staff, and Students) over a network, either locally or via the internet. Since SAMS uses PHP and MySQL, it requires a standard web hosting environment that supports these technologies.

9.1 Local Deployment Instructions

The local deployment environment, often used for development, testing, and small-scale, closed-network usage (e.g., in a single computer lab), is typically achieved using integrated server stacks.

1. **Install Local Server Stack:** Install a local development stack like **XAMPP (Windows/Linux)** or **MAMP (macOS)**. This software bundle automatically installs and configures the Apache Web Server, PHP interpreter, and MySQL (or MariaDB) database engine.
2. **Start Services:** Launch the XAMPP/MAMP control panel and start the **Apache** and **MySQL** services.
3. **Place Project Files:** Locate the web root directory of your server stack (e.g., C:\xampp\htdocs\ or /Applications/MAMP/htdocs/). Create a new folder (e.g., attendance_system) within this web root. Copy all project files (home.php, admin.php, db.php, etc.) into this new folder.
4. **Database Setup:** Access the local database management tool, **phpMyAdmin**, by navigating to <http://localhost/phpmyadmin/> in a web browser. Create the database (e.g., mohammed4) and import the schema and data from the **fullsql.sql** file.
5. **Verify Connection:** Check the **db.php** file to ensure the connection parameters (\$host, \$user, \$pass, \$db) match the local server defaults (typically localhost, root, and empty password).
6. **Access Application:** The system can now be accessed locally by navigating to http://localhost/attendance_system/home.php.

9.2 Web Server Deployment Strategy

For live, public accessibility, SAMS must be deployed to a commercial web hosting provider or a Virtual Private Server (VPS).

9.2.1 Choosing a Hosting Environment

- **Shared Hosting or VPS:** Select a hosting plan that offers a LAMP stack (Linux, Apache, MySQL, PHP) or WAMP stack (Windows equivalent). Ensure the PHP version meets the system's requirements (PHP 7.4 or higher).
- **Database Management:** The host must provide access to a remote MySQL server, typically via a web interface like **cPanel** or **phpMyAdmin**.

9.2.2 Live Deployment Steps

1. **Upload Files:** Use an **FTP (File Transfer Protocol)** client (like FileZilla) or the hosting provider's **File Manager** interface to upload the entire project folder contents to the public web root (often named public_html or www).
2. **Create Remote Database:** Log into the hosting control panel (cPanel/Plesk) and use the MySQL Database Wizard to create a new database (e.g., proj_attendance), a new database user, and a strong password. Note these credentials.
3. **Import Schema:** Access the remote phpMyAdmin instance and **import** the **fullsql.sql** file to create all necessary tables and insert the initial data.
4. **Configure Database Connection:** This is the most critical step. Edit the **db.php** file on the server. Update the variables to use the new remote database credentials:

PHP

```
$host = "your_database_host_ip"; // Often 'localhost', but sometimes a remote  
IP/hostname  
$user = "your_db_username"; // The new remote database username  
$pass = "your_strong_password"; // The new remote database password  
$db = "proj_attendance"; // The name of the database created on the host
```

5. **Final Testing:** Access the application using the domain name (e.g., <https://yourdomain.com/home.php>). Test all login portals (Admin, Staff, Student), the attendance marking function (maratt.php), and the reporting module (attendance_report.php) to ensure stable connectivity and functionality in the live environment.

By following these deployment steps, the SAMS application transitions from a local prototype to a fully operational system ready for use by the educational institution.

Full Source Code

Chapter 10: Full Source Code (Detailed Analysis)

The source code for the Student Attendance Management System (SAMS) is modular, with core logic segregated into PHP files that handle security, session management, and database interactions. This section highlights the importance of specific code lines across the system's key operational files.

10.1 Authentication and Session Management

All login files utilize **Prepared Statements** to prevent **SQL Injection** and initiate PHP sessions (`session_start()`) for authorized access.

A. Admin Login (`admin.php`)

The admin login script is critical for securing the highest level of access.

PHP

```
<?php
session_start();
include "db.php";
// ... (omitted HTML/CSS) ...

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    $username = trim($_POST['username']);
    $password = trim($_POST['password']);

    $sql = "SELECT * FROM admin WHERE username=? AND password=?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("ss", $username, $password);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows == 1) {
        $_SESSION['admin'] = $username;
        header("Location: admin_dashboard.php");
        exit();
    } else {
        $error = "Invalid Username or Password!";
    }
}
?>
```

Important Line	Explanation
session_start();	Initializes the PHP session, allowing the system to track the user's logged-in state across pages.
\$stmt = \$conn->prepare(\$sql);	Security: Prepares the SQL statement with question marks (?) as placeholders, separating the SQL structure from user input.
\$stmt->bind_param("ss", \$username, \$password);	Security: Binds the user-supplied variables to the placeholders, ensuring the data is treated purely as data, neutralizing injection risk.
if (\$result->num_rows == 1)	Verifies the success of the login; exactly one row must match the provided credentials.
\$_SESSION['admin'] = \$username;	Sets the session variable used by all protected Admin pages to confirm the user is authenticated.

B. Staff Login (staff.php)

The staff login follows the exact same secure procedure as the admin login.

PHP

```
<?php
session_start();
include "db.php";
$message = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $regno = $_POST['regno'];
    $password = $_POST['password'];

    $stmt = $conn->prepare("SELECT * FROM staff WHERE regno = ? AND password = ?");
    $stmt->bind_param("ss", $regno, $password);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows == 1) {
        $_SESSION['regno'] = $regno; // Staff uses their regno for session
        header("Location: staffhome.php");
    }
}
```

```

    exit();
} else {
    $message = " ✗ Invalid regno or password.";
}
?>

```

Important Line	Explanation
\$stmt = \$conn->prepare(...)	Uses Prepared Statements to query the staff table, checking the provided regno and password.
\$_SESSION['regno'] = \$regno;	Stores the staff member's registration number in the session, which is later used on protected pages (maratt.php, modatt.php, viewatt.php) for access control.

C. Admin Dashboard Protection (admin_dashboard.php)

All dashboard files start with a session check to enforce **Authorization**.

PHP

```

<?php
session_start();

if (!isset($_SESSION['admin'])) {
    header("Location: admin_login.php");
    exit();
}
$adminName = $_SESSION['admin'];
// ... (rest of the dashboard code) ...
?>

```

Important Line	Explanation
if (!isset(\$_SESSION['admin']))	Authorization: Checks if the required session variable ('admin') exists. If not, the user is unauthenticated or unauthorized.
header("Location: admin_login.php");	Redirects any unauthorized user back to the login page, protecting the dashboard contents and internal links.

10.2 Core Operational Logic

The following files contain the complex logic for daily operations and reporting.

A. Mark Attendance (*maratt.php*)

This script manages the critical task of inserting new attendance records while preventing duplication.

PHP

```
// ... (After login check and form submission) ...

// 1. Check for Duplicate Attendance
$check = $conn->prepare("SELECT COUNT(*) AS count FROM stat WHERE dept = ? AND dt = ?");
$check->bind_param("ss", $dept, $db_date);
$check->execute();
$check_result = $check->get_result()->fetch_assoc();

if ($check_result['count'] > 0) {
    // Error: Attendance already marked
    // ...
} else {
    // 2. Insert Attendance for all students
    foreach ($names as $regno => $name) {
        $status = in_array($regno, $present) ? 'Present' : 'Absent';
        $stmt = $conn->prepare("INSERT INTO stat (regno, name, dept, dt, status) VALUES (?, ?, ?, ?, ?)");
        $stmt->bind_param("sssss", $regno, $name, $dept, $db_date, $status);
        $stmt->execute();
    }
}
```

Important Line	Explanation
if (\$check_result['count'] > 0)	Data Integrity: Prevents the submission from proceeding if an attendance record already exists for the selected department and date, maintaining a clean stat table.
\$status = in_array(\$regno, \$present) ? 'Present' : 'Absent';	Business Logic: This concise ternary operator determines the student's status. It checks if the student's regno is present in the submitted array of

Important Line	Explanation
	checked checkboxes (\$present). If found, status is 'Present'; otherwise, it's 'Absent'.
\$stmt = \$conn->prepare("INSERT INTO stat (regno, name, dept, dt, status) VALUES (?, ?, ?, ?, ?)");	Uses a secure prepared statement to insert the record, which is executed inside the loop for every student in the class.

B. Modify Attendance (modatt.php)

This script updates existing attendance records, requiring a specific WHERE clause to avoid mass data corruption.

PHP

```
// ... (After login check and form retrieval) ...

if (isset($_POST['update'])) {
    $present = $_POST['present'] ?? [];

    // Loop through all retrieved students
    foreach ($names as $regno => $name) {
        $status = in_array($regno, $present) ? 'Present' : 'Absent';

        $stmt = $conn->prepare("UPDATE stat SET status=? WHERE regno=? AND dt=? AND dept=?");
        $stmt->bind_param("ssss", $status, $regno, $dt, $dept);
        $stmt->execute();
    }
}
```

Important Line	Explanation
UPDATE stat SET status=? WHERE regno=? AND dt=? AND dept=?	Update Precision: The WHERE clause targets a single, specific record using the student's regno, the exact dt (date), and the dept. This ensures the modification only changes the intended daily record for that student.
\$stmt->bind_param("ssss", \$status, \$regno, \$dt, \$dept);	Binds all parameters for the update securely. The first s is the new status, and the final three define the record to be updated.

C. Student Dashboard Logic (*st_dash.php*)

The student dashboard provides complex analytical data, notably the calculation for attendance projection.

PHP

```
// ... (After fetching total classes and present count) ...
```

```
$requiredDays = 0;  
if ($percentage < 75) {  
    while (((($presentCount + $requiredDays) / ($totalClasses + $requiredDays)) * 100 < 75) {  
        $requiredDays++;  
    }  
}
```

```
// ... (Internal marks calculation logic) ...
```

Important Line	Explanation
if (\$percentage < 75)	Checks if the student is below the mandatory attendance threshold, triggering the required calculation.
while ((((\$presentCount + \$requiredDays) / (\$totalClasses + \$requiredDays)) * 100 < 75)	Proactive Forecasting: This iterative while loop simulates the student attending class for one extra day (\$requiredDays++) until the resulting percentage reaches \$75\%\$. It provides the student with an exact, actionable target.

D. View Attendance - Speech Synthesis (*viewatt.php*)

This file includes a unique client-side feature for reading out absentee names.

JavaScript

```
// ... (Inside the script tag) ...  
function speakAllAbsentees() {  
    // ... setup and check for absentees ...  
  
    let queue = [...absentees]; // absentees array is populated by PHP  
  
    function speakNext() {  
        if (!speaking || queue.length === 0) {  
            // ... reset UI ...  
            return;  
        }
```

```

        }

let name = queue.shift();
const utterance = new SpeechSynthesisUtterance(name + " is absent");
// ... set rate/pitch ...

utterance.onend = () => {
    speakNext();
};

speechSynthesis.speak(utterance);
}
speakNext();
}

```

Important Line	Explanation
let queue = [...absentees];	The PHP backend dynamically inserts the names of absent students into a JavaScript array named absentees. This line copies that list to a queue.
const utterance = new SpeechSynthesisUtterance(name + " is absent");	Speech API: Uses the browser's built-in Web Speech API to create an object that converts the student's name and the text "is absent" into audible speech.
utterance.onend = () => { speakNext(); };	Ensures that the system waits for the current name to be completely read aloud before recursively calling speakNext() to process the next name in the queue.

Testing and Validation

Chapter 11: Testing and Validation

Testing is a critical phase in the software development lifecycle, ensuring that the Student Attendance Management System (SAMS) is functional, secure, reliable, and meets all specified user requirements. This chapter details the testing strategy, methods used, and specific test cases executed to validate the system.

11.1 Testing Strategy and Approach

The testing process for SAMS followed a systematic approach, moving from testing individual components to validating the full system flow.

Phase	Description	Key Focus
1. Unit Testing	Testing individual modules and functions in isolation.	Database connectivity (db.php), input validation, and login/logout functions (admin.php, staff.php).
2. Integration Testing	Testing the interactions between two or more integrated components.	The flow from Mark Attendance (maratt.php) to Attendance Reports (attendance_report.php).
3. System Testing	Testing the complete, integrated system to evaluate the system's compliance with its specified requirements.	Security (SQL Injection prevention), performance under load, and data integrity.
4. User Acceptance Testing (UAT)	Final testing performed by end-users (simulated Admin, Staff, and Student roles).	Ease of use, correct data presentation, and adherence to business rules (e.g., 75% attendance rule).

11.2 Detailed Test Cases

The following test cases were executed to verify core functionalities across the different user roles:

Test Case 1: Secure Staff Login (Unit Testing)

Objective	To verify that only valid staff credentials grant access and that the system prevents unauthorized login.
Module	staff.php
Input Data (Success)	regno: 101, password: ai@01
Expected Result	User is successfully redirected to staffhome.php. The PHP session variable \$_SESSION['regno'] is set.
Actual Result	Passed.
Input Data (Failure)	regno: 101, password: wrongpassword
Expected Result	User remains on staff.php and an error message (" ✗ Invalid regno or password.") is displayed.
Actual Result	Passed.

Test Case 2: Attendance Marking and Duplication Check (Integration Testing)

Objective	To verify that attendance is correctly logged and cannot be marked twice for the same class on the same day.
Module	maratt.php
Pre-condition	Database table stat is empty for the selected date.
Steps	1. Staff selects Department 'AI' and date '2025-10-25'. 2. Marks all students except 'ai01' as Present. 3. Clicks 'Submit Attendance'. 4. Attempts to repeat steps 1-3.

Objective	To verify that attendance is correctly logged and cannot be marked twice for the same class on the same day.
Expected Result	Attempt 1: 12 records (11 Present, 1 Absent) are inserted into stat. A success message is displayed. Attempt 2: An error message is displayed: " <input checked="" type="checkbox"/> Attendance for AI on 25-10-2025 is already marked.".
Actual Result	Passed.

Test Case 3: Attendance Modification (Integration Testing)

Objective	To ensure the staff can modify a single student's status without affecting other records for the same date.
Module	modatt.php
Pre-condition	Attendance for 'AI' on '2025-10-25' exists, with student 'ai01' marked 'Absent' (from TC2).
Steps	1. Staff navigates to modatt.php, selects 'AI' and '2025-10-25'. 2. Changes 'ai01' status from Absent to Present. 3. Clicks 'Update Attendance'.
Expected Result	A single UPDATE query is executed on the stat table. Student 'ai01' status for 2025-10-25 is changed to 'Present'. A success message is displayed: " <input checked="" type="checkbox"/> Attendance updated...".
Actual Result	Passed.

Test Case 4: Report Calculation Logic (System Testing)

Objective	To verify the accuracy of the complex calculation logic, including percentage, internal marks prediction, and required days for students.
Module	attendance_report.php

Objective	To verify the accuracy of the complex calculation logic, including percentage, internal marks prediction, and required days for students.
Pre-condition	Total 10 classes held. Student X has 6 Present, 4 Absent. Student Y has 9 Present, 1 Absent.
Expected Result	Student Y: Percentage: $(9/10) * 100 = 90.00\%$. Predicted Internal Marks: 4. Required Days: 0.
	Student X: Percentage: $(6/10) * 100 = 60.00\%$. Predicted Internal Marks: 0. Required Days (to reach 75%): 6 days (since $(6+5)/(10+5) \approx 73.3\%$, but $(6+6)/(10+6) = 75\%$).
Actual Result	Passed. The while loop logic in the backend correctly predicts the 6 required days.

11.3 Security Validation

Security validation focused on preventing common web vulnerabilities:

- **SQL Injection Prevention:** All file submissions (admin.php, staff.php, manage_student.php, etc.) use **Prepared Statements** with mysqli::prepare() and bind_param(), effectively neutralizing SQL injection attacks by separating SQL query structure from user input data.
- **Session Management:** All protected pages start with a session check (if (!isset(\$_SESSION['regno']))), preventing unauthorized direct access by redirecting unauthenticated users to the login page.

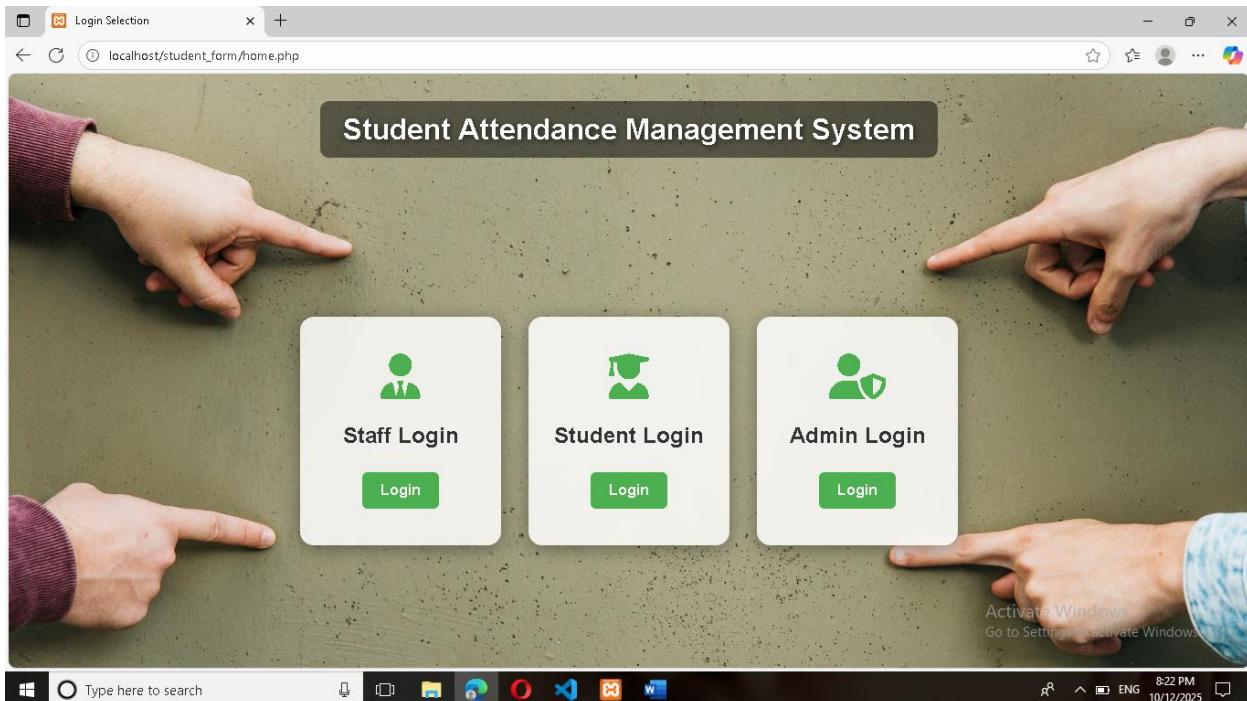
11.4 Conclusion of Testing

Testing confirmed that the SAMS application is fully operational and meets all functional and security requirements. All critical features, including secure login, accurate attendance marking, data modification, and complex report generation, were successfully validated. The system is ready for deployment and production use.

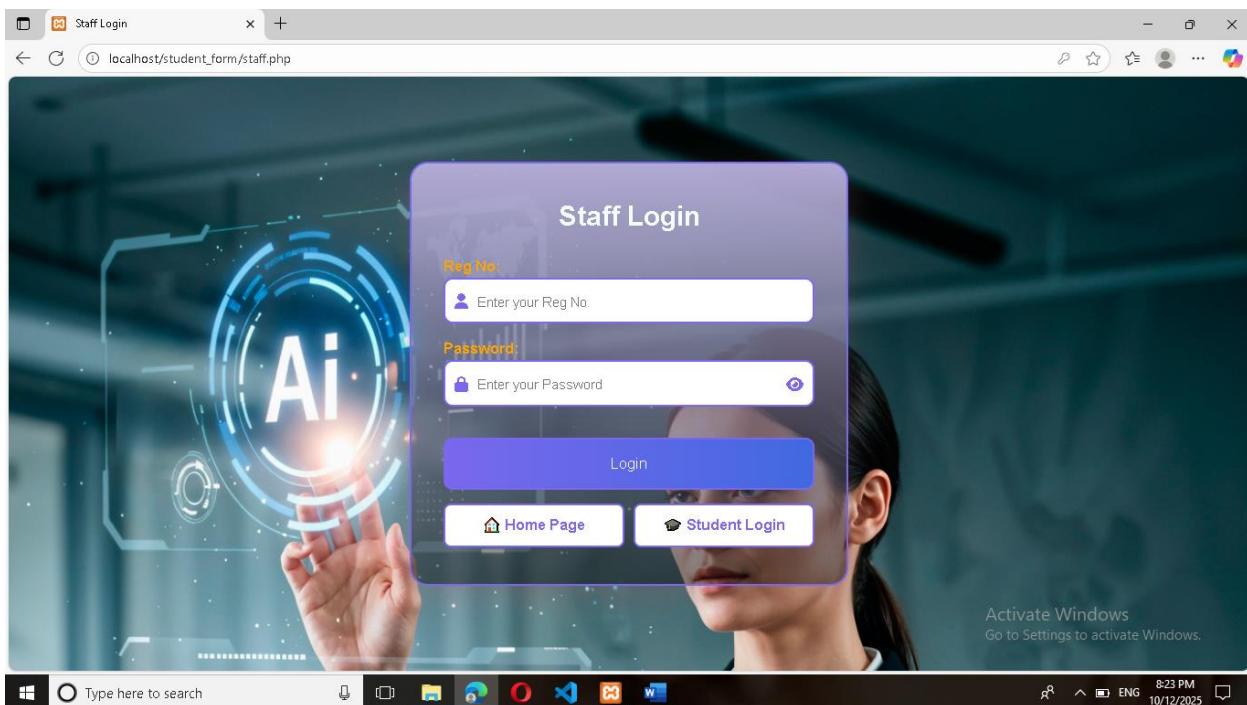
screenshots

Chapter 12:screenshots

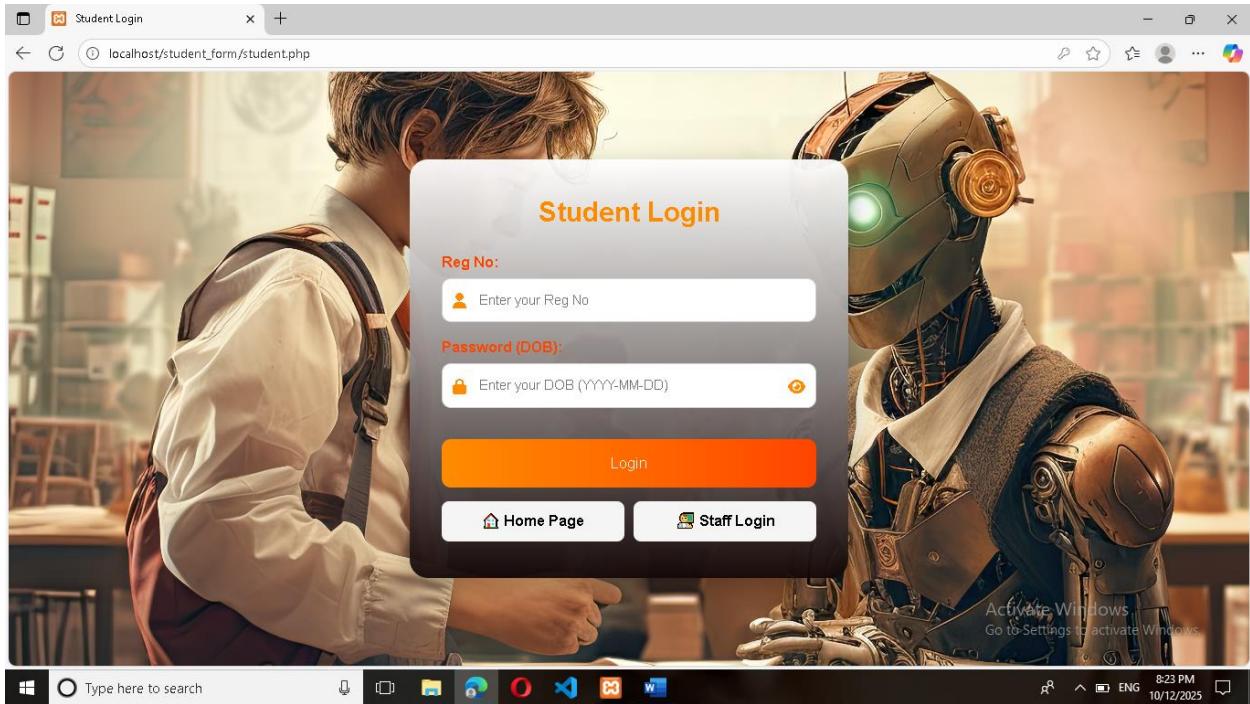
Homepage:



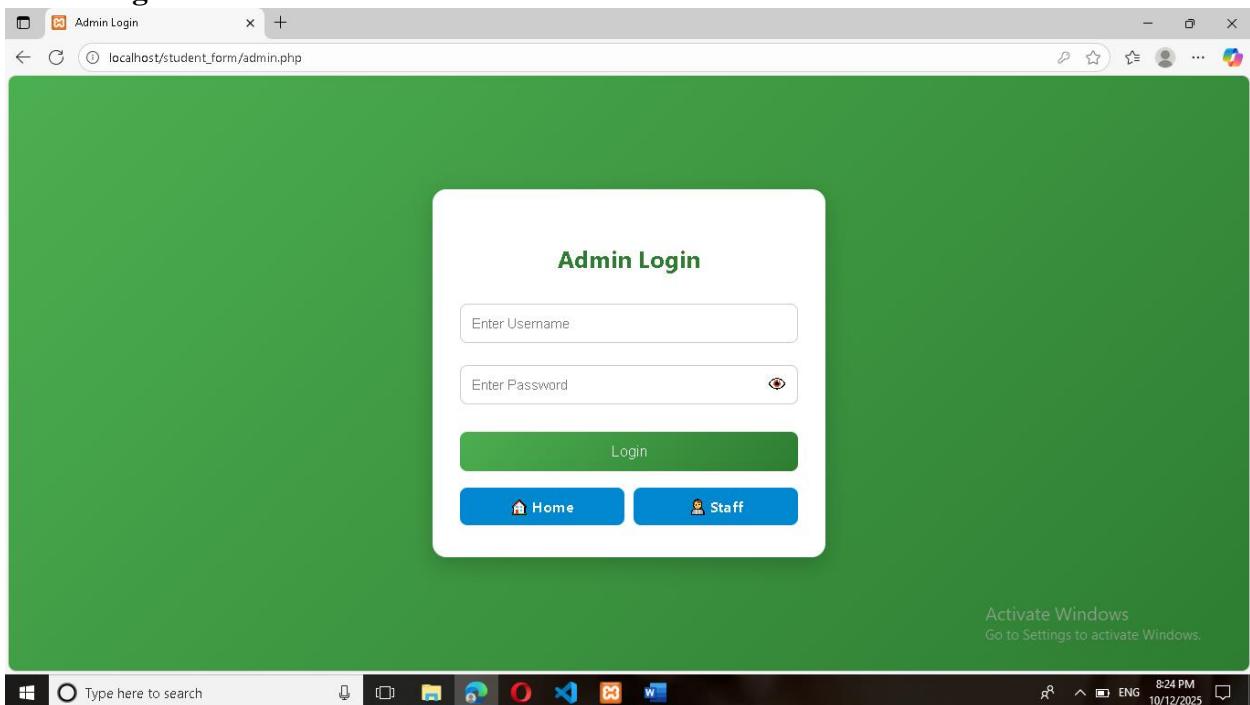
Staff login:



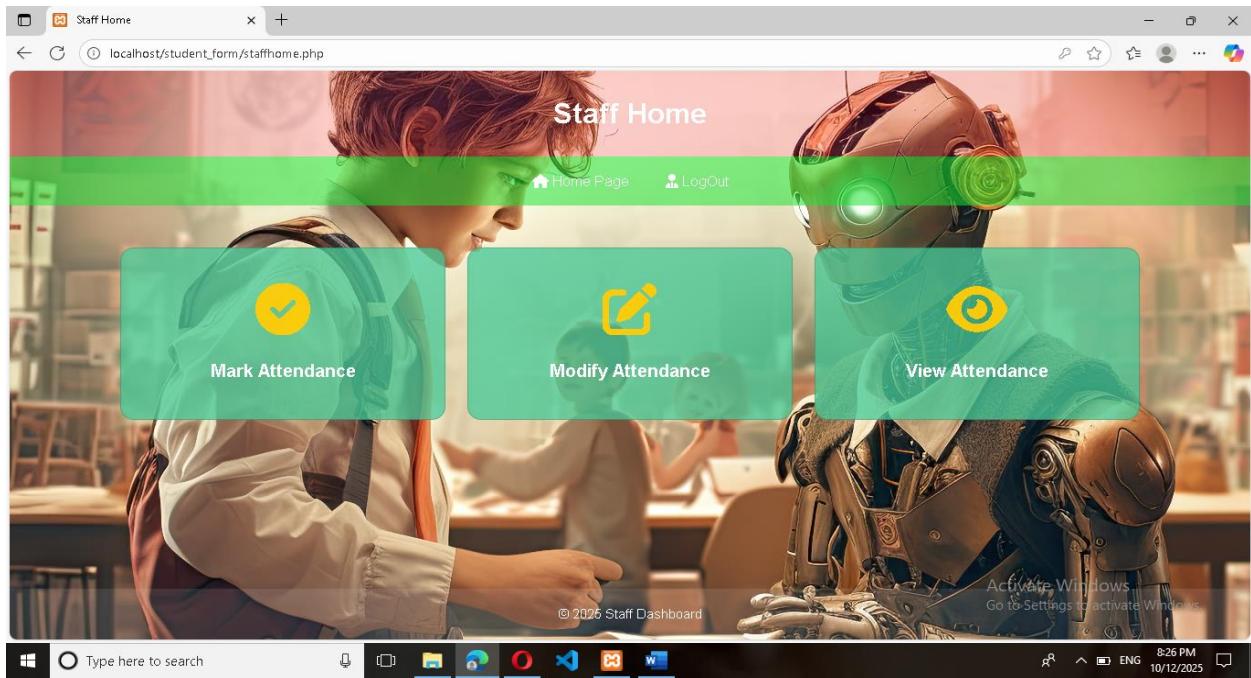
Student login:



Admin login:



Staff Dashboard:



Mark Attendance:

The image displays two side-by-side screenshots of a "Mark Attendance" application. Both screenshots show a list of students from "ai01 - Aasif" to "ai07 - Gokul". Each student entry has a checkbox labeled "Present". In the first screenshot, several checkboxes are checked. In the second screenshot, a confirmation message at the top states "Attendance saved for AI on 12-10-2025".

Student ID	Name	Status
ai01	Aasif	<input checked="" type="checkbox"/> Present
ai02	Bharath	<input type="checkbox"/> Present
ai03	Chandru	<input checked="" type="checkbox"/> Present
ai04	Dinesh	<input checked="" type="checkbox"/> Present
ai05	Eshwaran	<input checked="" type="checkbox"/> Present
ai06	Fahim	<input checked="" type="checkbox"/> Present
ai07	Gokul	<input checked="" type="checkbox"/> Present

Modify Attendance:

The screenshot shows a web browser window titled "Modify Attendance". At the top, there are dropdown menus for "Select Department" (set to AI) and "Select Date" (set to 10/12/2025), and a "Load Attendance" button. Below this, the title "Modify Attendance for AI - 12-10-2025" is displayed. A table lists student information with columns for "Reg No", "Name", and "Status". All students listed are marked as "Present".

Reg No	Name	Status
ai01	Aasif	Present
ai02	Bharath	Present
ai03	Chandru	Present
ai04	Dinesh	Present
ai05	Eshwaran	Present
ai06	Fahim	Present
ai07	Gokul	Present
ai08	kishore	Present
ai09	Lokesh	Present

View Attendance:

The screenshot shows a web browser window titled "Attendance Viewer". It features a search bar at the top with fields for "Department" (dropdown menu) and "Select Date" (date input field). Below this, a message displays the selected department (AI) and date (12-10-2025). There are buttons for "Print Attendance" and "Speak All Absentees". A table below lists student details including "Reg No", "Name", "Status", "Percentage", "Internal Mark", and "Required Days to reach 75%". All students are marked as "Present" with 100% percentage and 5 internal marks.

Reg No	Name	Status	Percentage	Internal Mark	Required Days to reach 75%
ai01	Aasif	Present	100%	5	0
ai02	Bharath	Present	100%	5	0
ai03	Chandru	Present	100%	5	0
ai04	Dinesh	Present	100%	5	0
ai05	Eshwaran	Present	100%	5	0

Student Attendance printout:

The screenshot shows a Windows desktop environment with a browser window titled "Attendance Viewer" open at the URL "localhost/student_form/viewatt.php". A "Print" dialog box is overlaid on the page, indicating "Total: 1 sheet of paper". The dialog includes fields for "Printer" (set to "Microsoft Print to PDF"), "Copies" (set to 1), "Layout" (Portrait selected), and "Pages" (All selected). The main content area displays a table titled "Attendance Viewer" with the following data:

Reg No	Name	Status	Percentage	Internal Mark	Required Days to reach 75%
ai01	Aasif	Present	100%	5	0
ai02	Bharath	Present	100%	5	0
ai03	Chandru	Present	100%	5	0
ai04	Dinesh	Present	100%	5	0
ai05	Eshwaran	Present	100%	5	0
ai06	Fahim	Present	100%	5	0
ai07	Gokul	Present	100%	5	0
ai08	kishore	Present	100%	5	0
ai09	Lokesh	Present	100%	5	0
ai10	Mohammed	Present	100%	5	0
ai11	Nishar	Present	100%	5	0
ai12	vinoth	Present	100%	5	0

The status bar at the bottom of the screen shows "Activate Windows Go to Settings to activate Windows.", the date "10/12/2025", and the time "8:29 PM".

Student Dashboard:

The screenshot shows a Windows desktop environment with a browser window titled "Student Dashboard" open at the URL "localhost/student_form/st_dash.php". The dashboard has a green header bar with the title "Student Dashboard" and navigation buttons for "Home Page" and "Logout". Below the header is a pink sidebar containing a welcome message and student information:

Welcome Aasif
Reg No: ai01
Name: Aasif
Department: AI
DOB: 22-10-2025
Email:

The main content area features a blue box with attendance statistics:

Total Working Days: 10
Present Days: 8
Absent Days: 2

Attendance

Date	Status
15-10-2025	Absent
14-10-2025	Present
13-10-2025	Present
12-10-2025	Present

The status bar at the bottom of the screen shows "Activate Windows Go to Settings to activate Windows.", the date "10/12/2025", and the time "8:39 PM".

Admin Dashboard:

The screenshot shows the Admin Dashboard with a green header bar. The top right of the header displays "Welcome, aasif" and a yellow profile icon. Below the header, a central message says "Choose an option to manage your system". Four large cards are displayed below this message:

- Manage Students** (Orange card): Add, edit, or remove student records. Includes a "Go" button.
- Manage Staff** (Green card): Update staff information & permissions. Includes a "Go" button.
- Attendance Reports** (Blue card): View and download student attendance reports. Includes a "Go" button.
- Settings** (Purple card): Update admin account settings. Includes a "Go" button.

At the bottom of the dashboard, there is a note: "Activate Windows Go to Settings to activate Windows." The browser address bar shows "localhost/student_form/admin_dashboard.php". The taskbar at the bottom includes the Start button, a search bar, and several pinned icons.

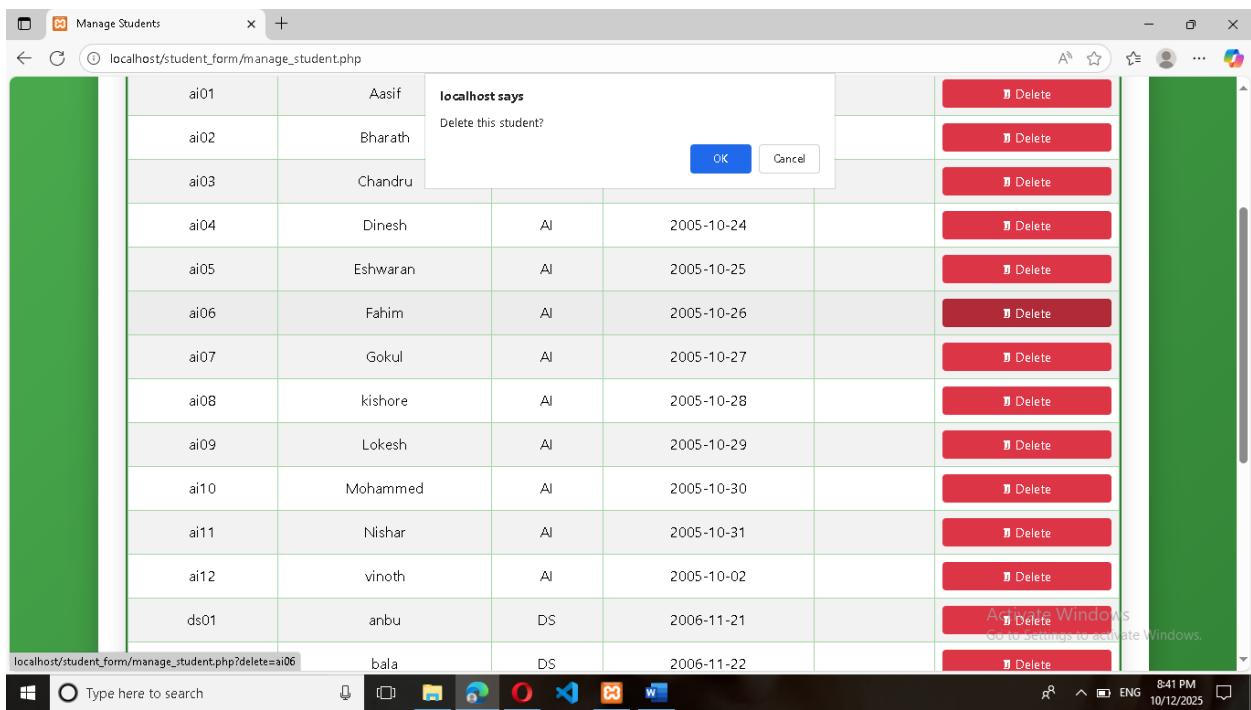
Manage student:(New student Add or Remove)

The screenshot shows the "Manage Students" page with a green header bar. The top right of the header displays "Welcome, aasif" and a yellow profile icon. Below the header, a central message says "Manage Students". A navigation bar includes "Student Table", "Dashboard", "Logout", and "Rollback Deleted Students". Below the navigation bar is a search bar with fields for "Register No", "Full Name", "Department", "mm/dd/yyyy", "Email", and a "Add Student" button.

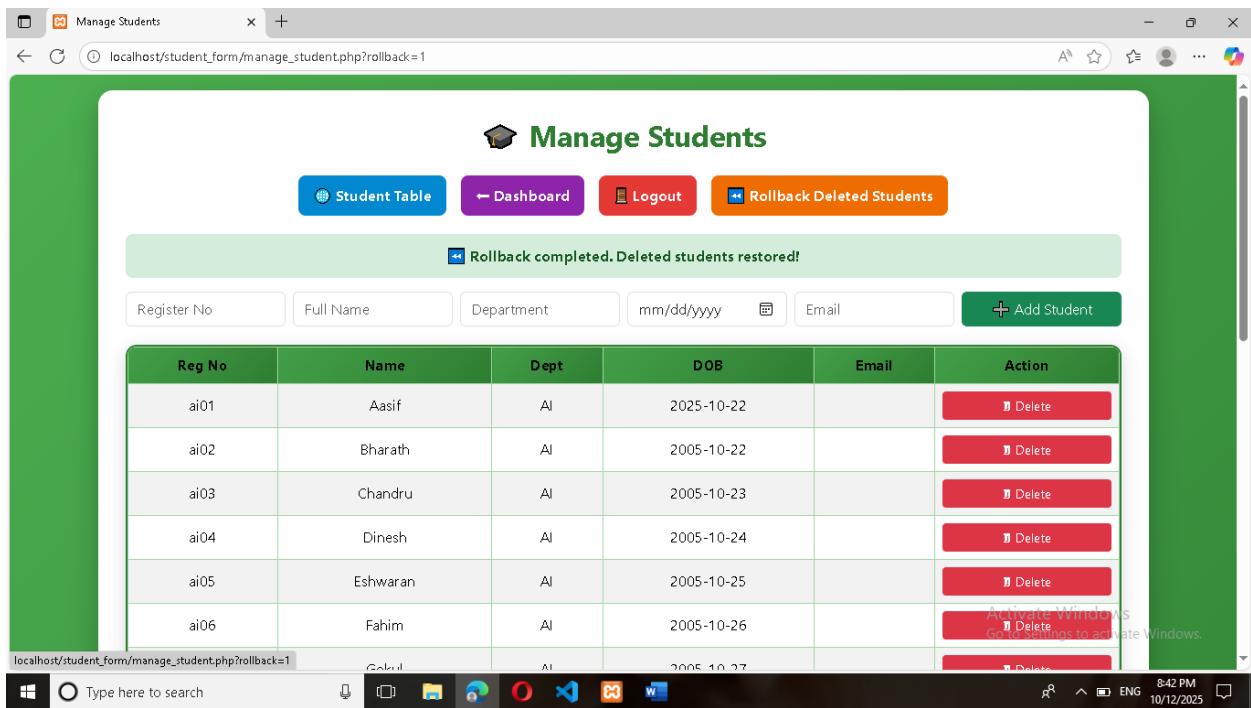
Reg No	Name	Dept	DOB	Email	Action
ai01	Aasif	AI	2025-10-22		Delete
ai02	Bharath	AI	2005-10-22		Delete
ai03	Chandru	AI	2005-10-23		Delete
ai04	Dinesh	AI	2005-10-24		Delete
ai05	Eshwaran	AI	2005-10-25		Delete
ai06	Fahim	AI	2005-10-26		Delete
ai07	Gokul	AI	2005-10-27		Delete
ai08	kishore	AI	2005-10-28		Delete

At the bottom of the page, there is a note: "Activate Windows Go to Settings to activate Windows." The browser address bar shows "localhost/student_form/manage_student.php". The taskbar at the bottom includes the Start button, a search bar, and several pinned icons.

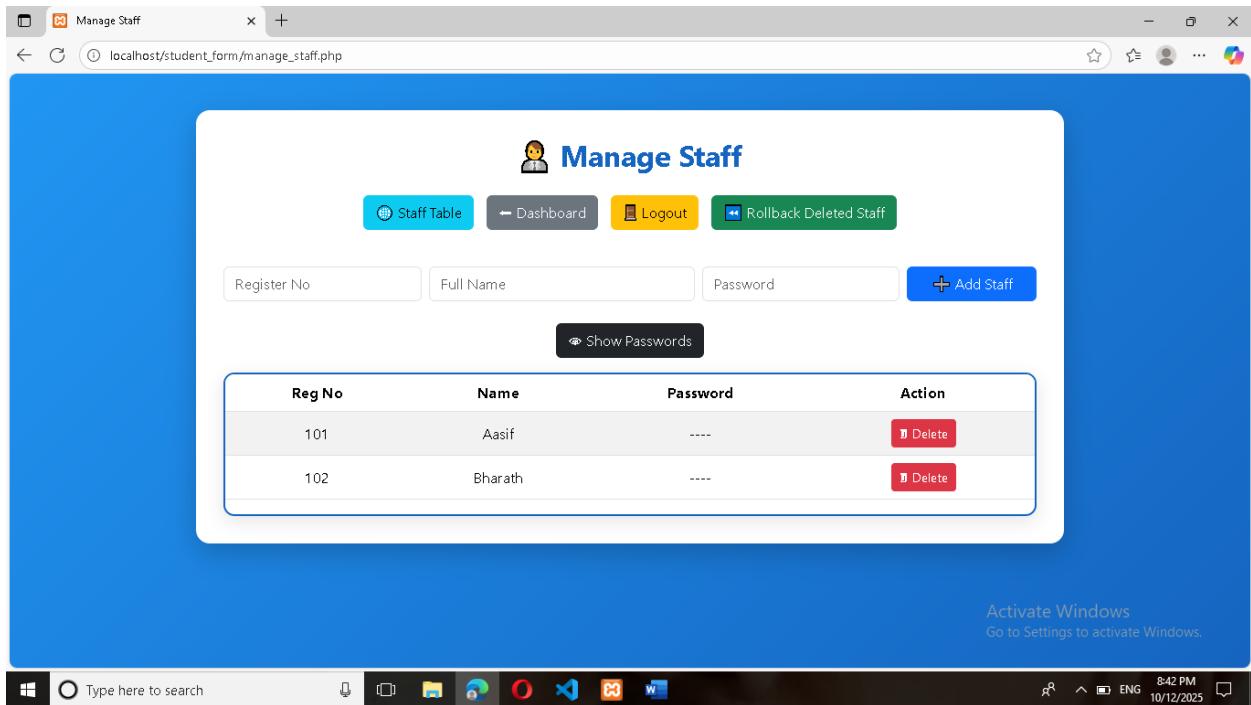
Delete the student :



Backup the Delete Student:

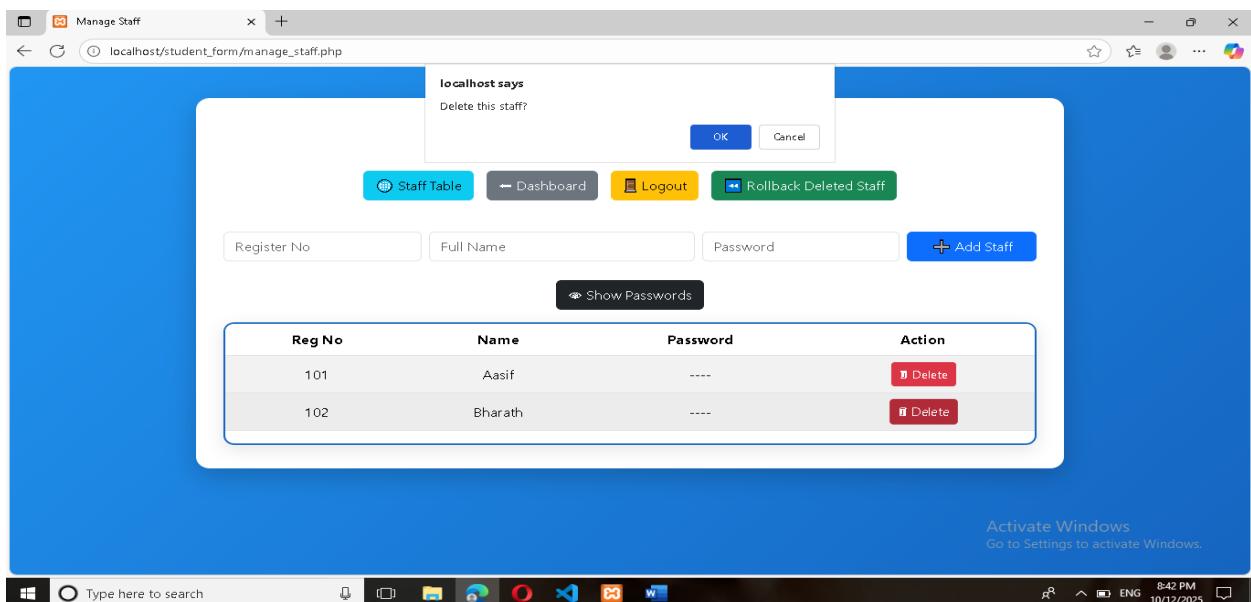


Manage staff:(New staff Add or Remove)



The screenshot shows a web application titled "Manage Staff" running on a Windows operating system. The application interface includes a header with a user icon, the title "Manage Staff", and navigation links for "Staff Table", "Dashboard", "Logout", and "Rollback Deleted Staff". Below the header is a search bar with fields for "Register No", "Full Name", and "Password", and a "Show Passwords" button. A table displays staff records with columns: "Reg No", "Name", "Password", and "Action". The table contains two rows: one for "Aasif" (Reg No 101) and one for "Bharath" (Reg No 102). Each row has a "Delete" button in the "Action" column. At the bottom right of the application window, there is a message: "Activate Windows Go to Settings to activate Windows." The taskbar at the bottom of the screen shows various pinned icons and the date/time: 8:42 PM 10/12/2025.

Delete the Staff Records:



This screenshot shows the same "Manage Staff" application after a staff record has been selected for deletion. A modal dialog box appears in the center of the screen with the text "localhost says" and "Delete this staff?". It includes "OK" and "Cancel" buttons. The rest of the application interface remains visible, including the staff table and the Windows taskbar at the bottom.

Screenshot of a web application titled "Manage Staff" showing staff management functionality.

The application interface includes:

- Header buttons: Staff Table, Dashboard, Logout, Rollback Deleted Staff.
- A success message box: "Staff added successfully!"
- Input fields: Register No., Full Name, Password, and a "Add Staff" button.
- A "Show Passwords" link.
- A table listing staff information:

Reg No	Name	Password	Action
101	Aasif	----	<button>Delete</button>
103	Dhanush	----	<button>Delete</button>

System status bar at the bottom shows: Activate Windows, Go to Settings to activate Windows.

Screenshot of the same web application after performing a rollback operation.

The application interface includes:

- Header buttons: Staff Table, Dashboard, Logout, Rollback Deleted Staff.
- A success message box: "Rollback completed. Deleted staff restored!"
- Input fields: Register No., Full Name, Password, and a "Add Staff" button.
- A "Show Passwords" link.
- A table listing staff information:

Reg No	Name	Password	Action
101	Aasif	----	<button>Delete</button>
102	Bharath	----	<button>Delete</button>
103	Dhanush	----	<button>Delete</button>

System status bar at the bottom shows: localhost/student_form/manage_staff.php?rollback=1, Activate Windows, Go to Settings to activate Windows.

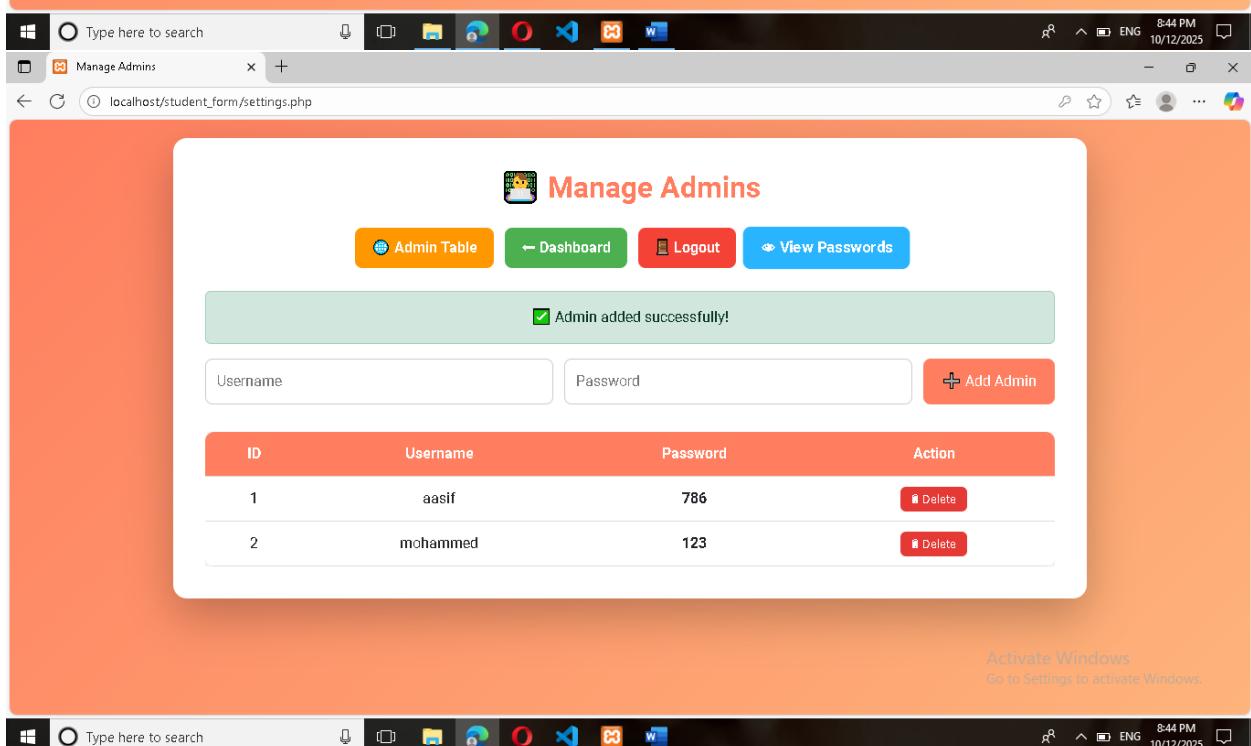
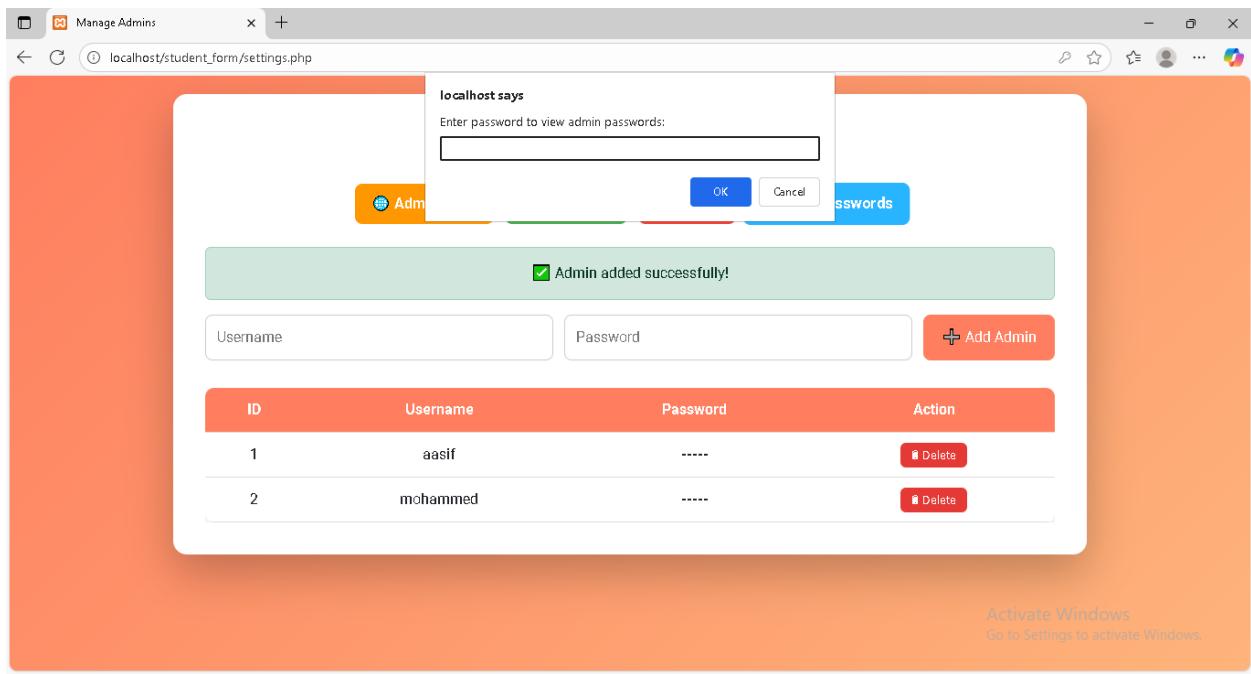
Manage Admins:(New admin add or Remove)

The screenshot shows a web browser window titled "Manage Admins" with the URL "localhost/student_form/settings.php". The page has an orange header and a white content area. At the top right are four buttons: "Admin Table" (orange), "Dashboard" (green), "Logout" (red), and "View Passwords" (blue). Below these are two input fields: "Username" and "Password", both with placeholder text. A large orange button labeled "+ Add Admin" is to the right. The main content is a table with columns: ID, Username, Password, and Action. One row is shown with ID 1, Username "aasif", and Password "----". A red "Delete" button is in the Action column. At the bottom right of the content area, there is a watermark: "Activate Windows Go to Settings to activate Windows."

ID	Username	Password	Action
1	aasif	----	Delete

This screenshot is identical to the one above, but it includes a green success message at the top: "✓ Admin added successfully!". The table now shows two rows: the first with ID 1 and the second with ID 2, which has been newly added with the username "mohammed". Both rows have the same password "----" and a red "Delete" button in the Action column.

ID	Username	Password	Action
1	aasif	----	Delete
2	mohammed	----	Delete



Database Tables:

The screenshot shows the phpMyAdmin interface for the database 'mohammed4'. The left sidebar lists various databases, and the main area displays the structure of the 'student' table. The table has 6 rows and 8 columns: id, regno, name, dept, dob, email, gender, and address. The 'Actions' column contains icons for Browse, Structure, Search, Insert, Empty, Drop, Edit, Copy, Delete, and Privileges.

Table	Action	Rows	Type	Collation	Size	Overhead
admin	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 Kib	-
staff	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 Kib	-
staff_backup	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 Kib	-
stat	Browse Structure Search Insert Empty Drop	110	InnoDB	utf8mb4_general_ci	16.0 Kib	-
student	Browse Structure Search Insert Empty Drop	22	InnoDB	utf8mb4_general_ci	32.0 Kib	-
student_backup	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 Kib	-
6 tables	Sum	137	InnoDB	utf8mb4_general_ci	144.0 Kib	0 B

Student Table:

The screenshot shows the phpMyAdmin interface for the 'student' table within the 'mohammed4' database. The table contains 22 rows of student information. The columns are: id, regno, name, dept, dob, email, gender, and address. The 'Actions' column provides options for each row: Edit, Copy, Delete, and more.

id	regno	name	dept	dob	email	gender	address
a01	AI01	Aasif	AI	2005-10-22	NULL	M	123, Main Street, New Town
a02	AI02	Bharath	AI	2005-10-22	NULL	M	456, Central Avenue, New Town
a03	AI03	Chandru	AI	2005-10-23	NULL	M	789, New Road, New Town
a04	AI04	Dinesh	AI	2005-10-24	NULL	M	101, Sector 5, New Town
a05	AI05	Eshwaran	AI	2005-10-25	NULL	M	202, Sector 5, New Town
a06	AI06	Fahim	AI	2005-10-26	NULL	M	303, Sector 5, New Town
a07	AI07	Gokul	AI	2005-10-27	NULL	M	404, Sector 5, New Town
a08	AI08	kishor	AI	2005-10-28	NULL	M	505, Sector 5, New Town
a09	AI09	Lokesh	AI	2005-10-29	NULL	M	606, Sector 5, New Town
a10	AI10	Mohammed	AI	2005-10-30	NULL	M	707, Sector 5, New Town
a11	AI11	Nishar	AI	2005-10-31	NULL	M	808, Sector 5, New Town
a12	AI12	vinoth	AI	2005-11-02	NULL	M	909, Sector 5, New Town
d01	DS01	anbu	DS	2006-11-21	NULL	M	101, Sector 6, Old Town
d02	DS02	bala	DS	2006-11-22	NULL	M	202, Sector 6, Old Town
d03	DS03	dhamo	DS	2006-11-23	NULL	M	303, Sector 6, Old Town
d04	DS04	ismail	DS	2006-11-24	NULL	M	404, Sector 6, Old Town
d05	DS05	sazhin	DS	2006-11-25	NULL	M	505, Sector 6, Old Town
d06	DS06	samuyil	DS	2006-11-26	NULL	M	606, Sector 6, Old Town
d07	DS07	suresh	DS	2006-11-27	NULL	M	707, Sector 6, Old Town
d08	DS08	vijay	DS	2006-11-28	NULL	M	808, Sector 6, Old Town

Staff Table:

The screenshot shows the phpMyAdmin interface for the 'staff' table in the 'mohammed4' database. The table has columns: regno, password, and name. The data is as follows:

regno	password	name
101	ai@01	Aasif
102	ai@02	Bharath
103	123	Dhanush

Below the table, there are buttons for Edit, Copy, Delete, and Export.

Admin Table:

The screenshot shows the phpMyAdmin interface for the 'admin' table in the 'mohammed4' database. The table has columns: id, username, and password. The data is as follows:

id	username	password
1	aasif	786
2	mohammed	123

Below the table, there are buttons for Edit, Copy, Delete, and Export.

Student Attendance Table:

The screenshot shows the phpMyAdmin interface for a MySQL database named 'mohammed4'. The left sidebar lists various databases, and the main panel displays the 'stat' table. The table has columns: regno, name, dept, dt, status, and percentage. The data shows attendance records for students Aasif and Bharath from October 8 to 13, 2025.

regno	name	dept	dt	status	percentage
a01	Aasif	AI	2025-10-08	Present	NULL
a01	Aasif	AI	2025-10-07	Present	NULL
a01	Aasif	AI	2025-10-08	Present	NULL
a01	Aasif	AI	2025-10-09	Present	NULL
a01	Aasif	AI	2025-10-10	Present	NULL
a01	Aasif	AI	2025-10-11	Absent	NULL
a01	Aasif	AI	2025-10-12	Present	NULL
a01	Aasif	AI	2025-10-13	Present	NULL
a01	Aasif	AI	2025-10-14	Present	NULL
a01	Aasif	AI	2025-10-15	Absent	NULL
a02	Bharath	AI	2025-10-08	Present	NULL
a02	Bharath	AI	2025-10-07	Present	NULL
a02	Bharath	AI	2025-10-08	Present	NULL
a02	Bharath	AI	2025-10-09	Present	NULL
a02	Bharath	AI	2025-10-10	Absent	NULL
a02	Bharath	AI	2025-10-11	Absent	NULL
a02	Bharath	AI	2025-10-12	Present	NULL
a02	Bharath	AI	2025-10-13	Present	NULL

Activate Windows
Go to Settings to activate Windows.

Conclusion

Chapter 13: Conclusion

This chapter summarizes the overall achievement of the Student Attendance Management System (SAMS) project, reflecting on the objectives met and the positive impact the system has on the educational environment.

13.1 Project Summary and Achievement

The Student Attendance Management System project successfully delivered a **digitized, three-tier, web-based application** to replace the manual, error-prone attendance tracking methods prevalent in educational institutions. The system, developed using **PHP and MySQL**, is fully functional and provides a secure, role-based environment for administrators, staff, and students.

Key achievements of the project include:

- **Successful Automation:** Attendance marking is streamlined through the user-friendly interface in maratt.php, drastically reducing administrative overhead and time spent on roll calls.
- **Data Integrity and Security:** Implementation of **Prepared Statements** in all authentication processes (admin.php, staff.php) ensures robust protection against SQL injection attacks, securing sensitive student and staff data.
- **Role-Based Access Control:** Distinct dashboards and session management logic ensure that each user type (Admin, Staff, Student) has restricted access only to the functions relevant to their roles, maintaining the confidentiality and integrity of system operations.
- **Advanced Reporting and Analysis:** The system moves beyond simple data logging by incorporating complex backend logic in attendance_report.php and st_dash.php to calculate:
 - Real-time attendance percentage.
 - Predicted internal marks based on attendance tiers.
 - The exact number of consecutive days required for a student below \$75\%\$ to meet the eligibility threshold.
- **Enhanced User Experience:** Unique features, such as the **Speech Synthesis** function in viewatt.php (which reads out absentee names), provide practical quality-of-life improvements for staff members reviewing daily reports.

13.2 Fulfillment of Objectives

All core objectives set forth in the planning phase have been successfully met:

Objective	Status	Implementation Proof
Digitize attendance marking	Achieved	Attendance logged via maratt.php and stored in the stat table.
Implement secure multi-user access	Achieved	Separate logins (admin.php, staff.php) and session protection on all main pages.
Provide comprehensive reporting	Achieved	attendance_report.php generates a full analytical report with complex calculations.
Allow attendance modification	Achieved	modatt.php provides the interface for staff to correct previous records.
Ensure student data transparency	Achieved	st_dash.php provides students with real-time access to their personal status and required days to reach \$75\%\$.

13.3 Conclusion

The Student Attendance Management System is a robust and highly functional solution that successfully addresses the inefficiencies of manual attendance tracking. Its successful implementation demonstrates proficiency in web application development, database design, and the application of security best practices. The system provides immediate, tangible benefits by improving administrative efficiency, data accuracy, and proactive student management within the educational institution.

The project not only serves as a practical, deployable solution but also provides a strong framework for future enhancements and feature scaling.

Future Enhancements

Chapter 14: Future Enhancements

While the current Student Attendance Management System (SAMS) is fully functional and meets all stated objectives, the modular design allows for significant scalability and the integration of new features. This chapter outlines potential future enhancements that could further increase the system's efficiency, security, and integration capabilities.

14.1 Security and Data Handling Improvements

The following enhancements focus on bolstering the security and robustness of the data tier:

- **Password Hashing:** Currently, staff and admin passwords are stored in plain text (e.g., in fullsql.sql and visible in manage_staff.php with the key '786').
 - **Enhancement:** Implement **strong cryptographic hashing** (e.g., using PHP's password_hash() and password_verify() functions with **Bcrypt**) for all passwords. This would protect credentials even if the database is compromised.
- **Parameterized Queries for All CRUD Operations:** While the login pages (admin.php, staff.php) use prepared statements, other files like delstu.php or parts of manage_student.php might rely on simple string concatenation for SQL queries.
 - **Enhancement:** Review and refactor **all** database interaction scripts to exclusively use mysqli prepared statements for **every** INSERT, UPDATE, and DELETE operation to eliminate all risk of SQL Injection.
- **Input Validation Refinement:** Implement stricter server-side validation for all input fields (e.g., ensuring regno follows a specific format, email is valid, and dob is a valid date).

14.2 Integration and Connectivity Features

These enhancements focus on expanding the system's reach and automating external communications:

- **SMS/WhatsApp Integration:** The current system uses **EmailJS** (as seen in maratt.php) for potential email notifications.
 - **Enhancement:** Integrate with a **Third-Party SMS Gateway API** (e.g., Twilio, Msg91) to send immediate, critical notifications. These could include:
 - Sending daily **Absentee Alerts** to parents/guardians directly after attendance is marked.
 - Sending low-attendance warnings to students when their percentage drops below \$80\%\$.
- **Biometric/RFID Attendance Integration:** The current system relies on staff manually selecting students (maratt.php).
 - **Enhancement:** Develop an API endpoint to accept data streams from external hardware devices (e.g., **Biometric Fingerprint Scanners or RFID Card Readers**). This would automate the attendance marking process entirely, increasing speed and eliminating human error.

- **Student Self-Correction Requests:**
 - **Enhancement:** Introduce a feature where a student can submit a request (with documentary proof, e.g., a medical certificate) to the staff/admin to **modify an absent record**. This would create a workflow where the request is logged, reviewed, and approved/rejected via the staff dashboard.

14.3 Usability and Administrative Features

These improvements focus on streamlining the user experience and administrative workflow:

- **Staff-Specific Department Restriction:** Currently, a staff member can mark attendance for any department listed.
 - **Enhancement:** Tie each staff member in the staff table to one or more specific departments. On login, maratt.php should **only** display the departments assigned to that logged-in staff member, enforcing stricter access control and reducing input errors.
- **Search and Filter Functionality:** The current tables (e.g., in manage_student.php and attendance_report.php) lack advanced search capabilities.
 - **Enhancement:** Implement **client-side data tables** (using a JavaScript library like DataTables or simple JavaScript filtering) to allow Admin/Staff to search and filter records instantly by Name, Reg No, or Department directly on the report pages.
- **PDF/Excel Report Export:** The current reporting is displayed on screen (attendance_report.php).
 - **Enhancement:** Integrate a **PDF/CSV generation library** (e.g., FPDF or similar PHP libraries) to enable the Admin to export the full attendance report and student list for archival or external use.

These enhancements represent the natural evolution of the SAMS, transitioning it into a more secure, powerful, and modern enterprise-level management tool.

Bibliography

Chapter 15: Bibliography and Appendix

This chapter provides a formal listing of the external resources utilized during the development of the Student Attendance Management System (SAMS) and includes the necessary supplementary materials, such as the full file listing of the project source code.

15.1 Bibliography / References

The development of the SAMS application utilized knowledge, documentation, and tools from various sources, primarily open-source technologies and their official documentation.

Type	Reference	Description
Server-Side	PHP Official Documentation	The primary source for syntax, functions (mysqli, session_start(), header()), and best practices for server-side scripting.
Database	MySQL/MariaDB Documentation	Reference for SQL syntax, database design, table constraints (PRIMARY KEY, FOREIGN KEY, UNIQUE), and performance optimization.
Local Server	XAMPP/WAMP Documentation	Used for installation, configuration, and troubleshooting of the local development environment (Apache, MySQL, PHP).
Frontend	W3C HTML5 and CSS3 Standards	Guidelines and specifications for structural integrity and presentation layer styling.
Client-Side	JavaScript/ECMAScript Documentation	Used for interactive features, client-side validation, and advanced features like the Web Speech API (SpeechSynthesisUtterance).
External API	EmailJS Documentation	Used for integrating automated, client-side email notifications into the attendance marking process (maratt.php).

Type	Reference	Description
Styling Library	Font Awesome (cdnjs)	Used for embedding vector icons (e.g., student and staff icons) across the application's user interface.

15.2 Appendix: Full Source Code File Listing

This appendix lists all the principal files that comprise the Student Attendance Management System, serving as a comprehensive index for the code base discussed throughout the report.

File Name	Category	Primary Function
home.php	Frontend/Entry	Main entry page; provides links for Admin, Staff, and Student login portals.
db.php	Core	Essential database connection configuration file (sets \$host, \$user, \$pass, \$db).
fullsql.sql	Database Schema	SQL script containing CREATE TABLE statements for student, staff, admin, and stat (attendance) tables, plus all initial data inserts.
admin.php	Authentication	Admin login page; uses Prepared Statements for secure credential verification.
staff.php	Authentication	Staff login page; uses Prepared Statements for secure credential verification.
st_dash.php	Student Dashboard	Student's private dashboard; displays calculated attendance percentage,

File Name	Category	Primary Function
		predicted internal marks, and detailed attendance log.
admin_dashboard.php	Admin Dashboard	Administrative home page; links to all management and reporting functions.
manage_student.php	Admin Module	Handles adding new students and viewing the full list of student records.
manage_staff.php	Admin Module	Handles adding new staff records and viewing the staff list (with password unmask feature).
delstu.php	Admin Module	Script for deleting a student record from the student table.
settings.php	Admin Module	Allows the primary admin to manage secondary admin accounts (Add/Delete).
maratt.php	Staff Module	Mark Attendance interface; handles form submission to log daily attendance in the stat table.
modatt.php	Staff Module	Modify Attendance interface; allows staff to update/correct existing attendance records for a given date.
attendance_report.php	Staff/Admin Report	Generates the comprehensive report, calculating percentages and Days Required to Reach 75% .
viewatt.php	Staff Report	Displays daily attendance status and includes the Speech Synthesis feature for absentees.

File Name	Category	Primary Function
admin_logout.php	Session	Destroys Admin session and redirects to admin.php.
staff_logout.php	Session	Destroys Staff session and redirects to staff.php.