RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

ATTENDANCE

MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

AKSHAYAA S	**231801007**
AMALA ENCILIN T	**231801009**
ARCHANA R M	**231801011**

In partial fulfillment for the award of the degree of

BACHELOR OF

TECHNOLOGY IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105
2024 - 2025

ABSTRACT

Efficient management of student attendance is a critical task for educational institutions, often requiring systematic tracking and easy access to attendance records. This project focuses on developing a Student Attendance Management System, leveraging a database-backed application to streamline this process.

The primary goal of this project is to provide a user-friendly interface for teachers and students to manage and view attendance data. The system features a role-based login mechanism, where teachers can mark attendance and add new students, while students can view their personal attendance records. This ensures a clear separation of functionalities based on user roles.

The project is implemented using a combination of technologies, including Python with Tkinter for the graphical user interface, and MySQL as the backend relational database for securely storing user credentials and attendance records. CRUD (Create, Read, Update, Delete) operations are performed through the integration of Python and MySQL, ensuring robust and scalable data management.

The application highlights the use of core database principles, GUI programming, and efficient role-based access control. It offers intuitive navigation for teachers to manage student data and for students to stay informed about their attendance status.

The objective of this project is to develop a basic attendance management system that simplifies record-keeping and provides an interactive, secure, and efficient platform for educational institutions to handle attendance tracking. It serves as a practical implementation of database and GUI programming principles, with the potential for future enhancements such as performance analytics and multi-user support.

TABLE OF CONTENTS

1. INTRODUCTION

2. SURVEY OF TECHNOLOGIES

3. REQUIREMENTS AND ANALYSIS

I. INTRODUCTION

1.1 INTRODUCTION

Managing student attendance is a fundamental aspect of educational institutions, yet it can be a time-consuming and error-prone task when done manually. To address this challenge, the Student Attendance Management System provides a digital solution that automates attendance tracking and management, ensuring accuracy, efficiency, and ease of use for both students and teachers.

This system allows students and teachers to interact seamlessly through a user-friendly interface. Students can log in to view their attendance records, while teachers have access to tools for marking attendance, adding new students, and managing records. With role-based access, the system ensures that each user's functionality is tailored to their specific needs, promoting efficiency and security.

The application utilizes a combination of Python with Tkinter for a graphical user interface and MySQL as the backend database for reliable data storage. Teachers can mark attendance interactively and provide real-time updates, while students can check their attendance status with ease. The system also includes robust validation mechanisms to maintain data integrity and prevent unauthorized access.

One of the key features of this system is its simplicity and scalability, making it suitable for institutions of varying sizes. By replacing manual record-keeping with a digitized system, this project aims to reduce administrative overhead and enhance productivity.

This project demonstrates the practical implementation of database management principles and GUI programming, offering a scalable and efficient platform for managing student attendance. Future enhancements such as analytics, reporting, and integration with broader school management systems can further expand its capabilities.

1.2 OBJECTIVES

Primary Objectives

1. Automate Attendance Tracking: Provide a digital platform for efficient and error-free attendance marking.
2. Role-Based Access Control: Ensure seamless functionality for both teachers and students by tailoring features to user roles.
3. Simplify Record Retrieval: Enable students to view their attendance records in a user-friendly format.
4. Streamline Data Management: Offer teachers tools to manage attendance and student records effectively.

Educational Objectives

1. Improve Administrative Efficiency: Minimize manual record-keeping efforts, reducing errors and saving time.
2. Enhance Engagement: Empower students to track their attendance and stay informed.
3. Promote Scalability: Lay the foundation for integrating additional modules, such as analytics or performance tracking.
4. Strengthen Security: Protect user data through robust authentication and database practices.

1.3 **Modules**

Admin Module (Teacher Role)

- Login & Dashboard: Secure login system with a dedicated teacher interface for attendance and student management.
- Attendance Management:
  - Mark attendance for students efficiently.
  - View attendance records for any student.
- Student Management:
  - Add new students with unique credentials.
  - Prevent duplicate entries by verifying usernames.
- Role Assignment: Assign specific roles (e.g., student, teacher) to ensure role-based access.

Student Experience Hub (Student Role)

- Login & Registration: Secure access for students to view attendance records.
- Attendance Overview:
  - Display attendance data in an organized, user-friendly format.
  - Show detailed records by date and status (e.g., Present, Absent).

Data Module

- User Database:
  - Store and manage user credentials and roles securely.
  - Ensure data integrity and prevent unauthorized access.
- Attendance Records:
  - Maintain a comprehensive log of attendance data for each student.
  - Enable quick retrieval and analysis of attendance trends.

Security Module

- Authentication System:
  - Validate credentials before granting access.
  - Prevent unauthorized access with strong password requirements.
- Database Security:
  - Safeguard attendance and user data using secure database operations.
- Session Management: Ensure secure interactions between the GUI and the database during user activity.

These modules clearly define the core functionality of the project, ensuring an organized, efficient, and secure attendance management system.

II. SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

2.2.1 Python

Python is a high-level, interpreted programming language known for its readability and simplicity. It is widely used in developing applications due to its rich library ecosystem and compatibility with various platforms. In this project, Python serves as the backbone for implementing the logic of the Student Attendance Management System. The integration of Tkinter, a GUI toolkit, enables the creation of an intuitive and user-friendly interface. Additionally, Python’s ability to interact seamlessly with MySQL ensures efficient database management and communication.

2.2.2 MySQL

MySQL is an open-source Relational Database Management System (RDBMS) that organizes data into tables, enabling structured data storage and retrieval. MySQL uses Structured Query Language (SQL) to perform operations like creating, modifying, and extracting data. It plays a critical role in this project by maintaining user credentials, attendance records, and other vital data securely and efficiently. MySQL ensures data integrity and supports real-time updates, which is essential for a reliable attendance management system.

2.2.3 Tkinter (GUI Framework)

Tkinter is Python's standard library for creating graphical user interfaces (GUIs). It provides pre-built widgets such as buttons, labels, and text entry fields that are essential for designing an interactive application. In this project, Tkinter is used to create a visually appealing and responsive interface that enables users to log in, view records, and manage attendance with ease. Tkinter's seamless integration with Python enhances the application’s functionality and user experience.

III. REQUIREMENTS AND ANALYSIS

User Requirements

The system requirements for the **Attendance Management System** focus on managing user accounts (students and teachers), enabling students to view their attendance records, and allowing teachers to mark attendance for students. The system should also provide functionality for adding new students and managing attendance data. Users (students and teachers) should be able to interact with the application through an easy-to-use graphical user interface (GUI) built using **Tkinter**.

Key Features:

- Login and user authentication (student and teacher roles).
- Students can view their attendance records.
- Teachers can mark attendance for students.
- Teachers can add new student accounts.
- Data is stored and retrieved from a **MySQL** database.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Software Requirements

- **Operating System**: Windows 10 or higher
- **Programming Language**: Python 3.x
- **Frontend**: Tkinter (for GUI)
- **Backend**: MySQL (for database management)
- **Database Management System (DBMS)**: MySQL 8.0 or higher
- **Python Libraries**: mysql-connector, tkinter
- **IDE/Editor**: Eclipse, PyCharm, or any Python-compatible IDE
- **Database Backup System**: MySQL Backup tools for data integrity

Hardware Requirements

- **System Type**: Desktop PC or Laptop
- **Operating System**: Windows 10 or higher
- **Processor**: Intel® Core™ i3-6006U CPU @ 2.00GHz or higher
- **RAM**: 4.00 GB RAM or higher

- **Storage**: 100 MB of available disk space for application and MySQL database files
- **Monitor Resolution**: 1024 x 768 pixels or higher
- **Input Devices**: Keyboard and Mouse

3.1 DATA DICTIONARY

1. Users Table

Table Name: users

Description:

This table stores user information (students and teachers) for login purposes. It contains fields to differentiate between different user roles (e.g., student, teacher). The user credentials (username and password) are stored here to authenticate users during login.

Column Name	**Data Type**	**Description**
id	INT AUTO_INCREMENT	Unique identifier for each user (Primary Key)
username	VARCHAR(50)	The username chosen by the user (Unique)
password	VARCHAR(255)	The password associated with the username
role	ENUM('student', 'teacher')	The role of the user (either 'student' or 'teacher')
email	VARCHAR(100)	Email address of the user

Usage:

- id is used to uniquely identify each user.
- username and password are used for user login.
- role defines if the user is a student or a teacher.
- email helps in sending notifications related to user registration, order status, etc.

2. Attendance Table

Table Name: attendance

Description:

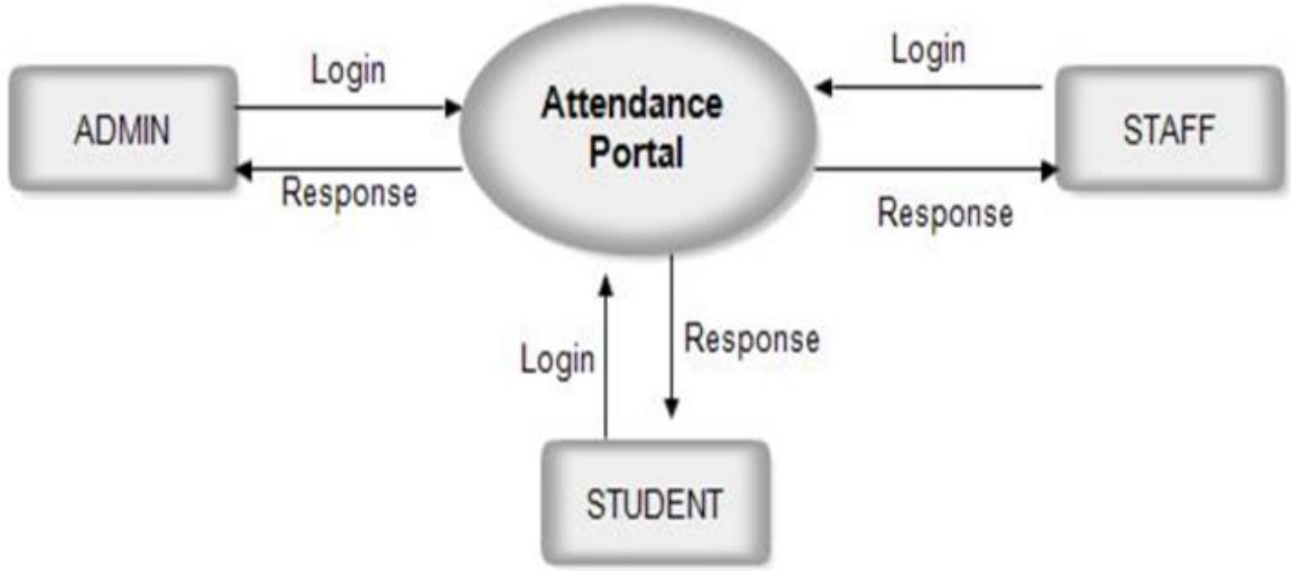
This table stores the attendance records of students. It tracks the student’s attendance status for each day, allowing teachers to mark attendance as either present or absent.

Column Name	**Data Type**	**Description**
id	INT AUTO_INCREMENT	Unique identifier for each attendance record (Primary Key)
student_id	INT	Foreign Key referencing id from the users table (student)
date	DATE	The date of the attendance record
status	ENUM('Present', 'Absent')	The attendance status for the student (either Present or Absent)

Usage:

- student_id links each attendance record to a specific student from the users table.
- date represents the date the attendance is recorded.
- status reflects the student’s attendance (either 'Present' or 'Absent').

ER DIAGRAM :


Login
Login
ADMIN
Attendance Portal
STAFF
Response
Response
Response
Login
STUDENT

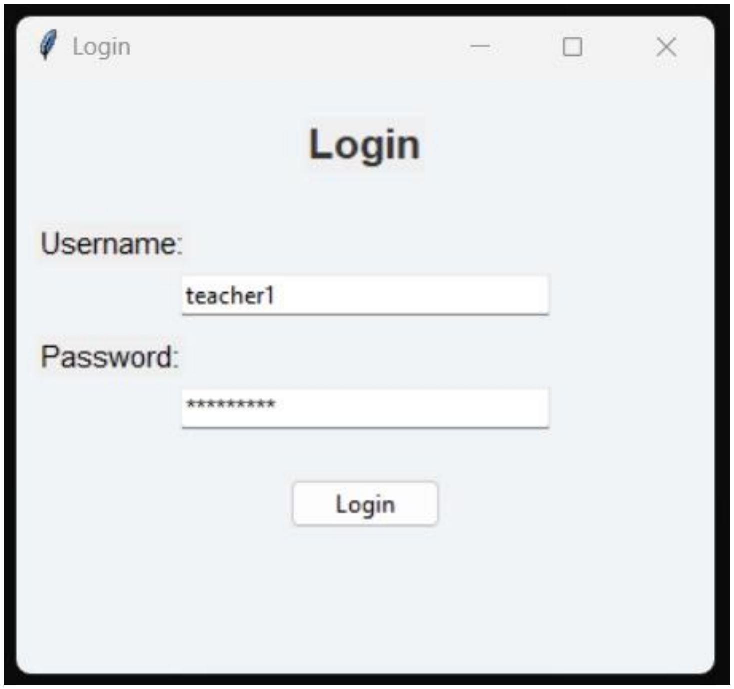
PROGRAM CODE:

```
import tkinter as tk
from tkinter import ttk, messagebox
import mysql.connector
from datetime import date

# Connect to the MySQL database
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Akshayaa2005@",
    database="school_db"
)

cursor = db.cursor()

# Global variables for entry fields
entry_new_student_username = None
entry_new_student_password = None

# Login function
def login():
    username = entry_username.get()
    password = entry_password.get()

    cursor.execute("SELECT id, role FROM users WHERE username=%s AND password=%s",
(username, password))
    result = cursor.fetchone()

    if result:
        user_id, role = result
        if role == 'student':
            open_student_page(user_id)
        elif role == 'teacher':
            open_teacher_page()
    else:
        messagebox.showerror("Error", "Invalid username or password")

# Open Student Page
def open_student_page(student_id):
    student_window = tk.Toplevel(root)
    student_window.title("Student Page")
    student_window.geometry("400x300")
    student_window.configure(bg="#f0f4f7")

    ttk.Label(student_window, text="Attendance Records", font=("Helvetica", 14, "bold"),
foreground="#333333").pack(pady=10)
```

```
    # Retrieve attendance records
    cursor.execute("SELECT date, status FROM attendance WHERE student_id=%s", (student_id,))
    records = cursor.fetchall()

    # Display attendance records in a table-like format
    frame = ttk.Frame(student_window)
    frame.pack(pady=10)
    if records:
        for record in records:
            ttk.Label(frame, text=f"Date: {record[0]}, Status: {record[1]}", font=("Helvetica", 11)).pack(anchor="w", padx=10)
    else:
        ttk.Label(student_window, text="No attendance records found.", font=("Helvetica", 11), foreground="gray").pack()

# Open Teacher Page
def open_teacher_page():
    global entry_new_student_username, entry_new_student_password  # Declare as global
    teacher_window = tk.Toplevel(root)
    teacher_window.title("Teacher Page")
    teacher_window.geometry("400x500")
    teacher_window.configure(bg="#f0f4f7")

    # Mark Attendance Section
    ttk.Label(teacher_window, text="Mark Attendance", font=("Helvetica", 14, "bold"), foreground="#333333").pack(pady=(10, 5))

    ttk.Label(teacher_window, text="Student ID:", font=("Helvetica", 11)).pack(anchor="w", padx=10, pady=(5, 0))
    entry_student_id = ttk.Entry(teacher_window, width=30)
    entry_student_id.pack(pady=5, padx=10)

    ttk.Label(teacher_window, text="Status (Present/Absent):", font=("Helvetica", 11)).pack(anchor="w", padx=10, pady=(5, 0))

    # Radio Buttons for Attendance Status
    attendance_status = tk.StringVar()
    attendance_status.set("Absent")  # default value

    ttk.Radiobutton(teacher_window, text="Present", variable=attendance_status, value="Present").pack(pady=5)
    ttk.Radiobutton(teacher_window, text="Absent", variable=attendance_status, value="Absent").pack(pady=5)

    # Add the Submit button to mark attendance
    ttk.Button(teacher_window, text="Submit", command=mark_attendance).pack(pady=10)
    # Separator line
    ttk.Separator(teacher_window, orient="horizontal").pack(fill="x", pady=10)

    # New Student Section
    ttk.Label(teacher_window, text="Add New Student", font=("Helvetica", 14, "bold"),
foreground="#333333").pack(pady=10)

    ttk.Label(teacher_window, text="Username:", font=("Helvetica", 11)).pack(anchor="w", padx=10,
pady=(5, 0))
    entry_new_student_username = ttk.Entry(teacher_window, width=30)
    entry_new_student_username.pack(pady=5, padx=10)

    ttk.Label(teacher_window, text="Password:", font=("Helvetica", 11)).pack(anchor="w", padx=10,
pady=(5, 0))
    entry_new_student_password = ttk.Entry(teacher_window, width=30, show="*")
    entry_new_student_password.pack(pady=5, padx=10)

    # Button to add the new student
    ttk.Button(teacher_window, text="Add Student", command=add_student).pack(pady=10)

def mark_attendance():
    student_id = entry_student_id.get()
    status = attendance_status.get()
    today = date.today()

    # Check if the student_id exists in the users table with role 'student'
    cursor.execute("SELECT id FROM users WHERE id = %s AND role = 'student'", (student_id,))
    if cursor.fetchone() is None:
        messagebox.showerror("Error", "Student ID does not exist or is not a student")
        return

    # Insert the attendance record if student_id is valid
    try:
        cursor.execute("INSERT INTO attendance (student_id, date, status) VALUES (%s, %s, %s)",
                       (student_id, today, status))
        db.commit()
        messagebox.showinfo("Success", "Attendance marked successfully")
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", str(err))

def add_student():
    username = entry_new_student_username.get()
    password = entry_new_student_password.get()
    role = 'student'  # Since we're adding a new student, set role to 'student'

    # Check if username already exists
    cursor.execute("SELECT id FROM users WHERE username = %s", (username,))
    if cursor.fetchone() is not None:
        messagebox.showerror("Error", "Username already exists. Please choose a different username.")
        return

    # Insert the new student record
    try:
        cursor.execute("INSERT INTO users (username, password, role) VALUES (%s, %s, %s)",
                       (username, password, role))
        db.commit()
        messagebox.showinfo("Success", "New student added successfully")
        # Clear entry fields after adding
        entry_new_student_username.delete(0, tk.END)
        entry_new_student_password.delete(0, tk.END)
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", str(err))

# Main Window
root = tk.Tk()
root.title("Login")
root.geometry("350x300")
root.configure(bg="#f0f4f7")

ttk.Label(root, text="Login", font=("Helvetica", 16, "bold"), foreground="#333333").pack(pady=20)

ttk.Label(root, text="Username:", font=("Helvetica", 11)).pack(anchor="w", padx=10, pady=(5, 0))
entry_username = ttk.Entry(root, width=30)
entry_username.pack(pady=5, padx=10)

ttk.Label(root, text="Password:", font=("Helvetica", 11)).pack(anchor="w", padx=10, pady=(5, 0))
entry_password = ttk.Entry(root, width=30, show="*")
entry_password.pack(pady=5, padx=10)

ttk.Button(root, text="Login", command=login).pack(pady=20)

root.mainloop()

# Close the database connection when done
cursor.close()
db.close()
```

RESULT AND DISCUSSION

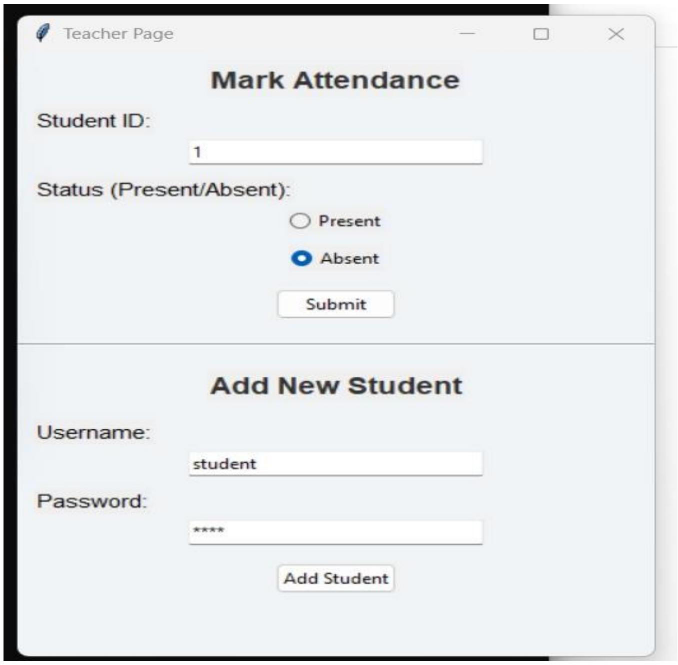
LOGIN PAGE



Username: teacher1

Password: *********

TO ADD AND MARK THE ATTENDANCE OF THE STUDENT



Mark Attendance

Student ID: 1

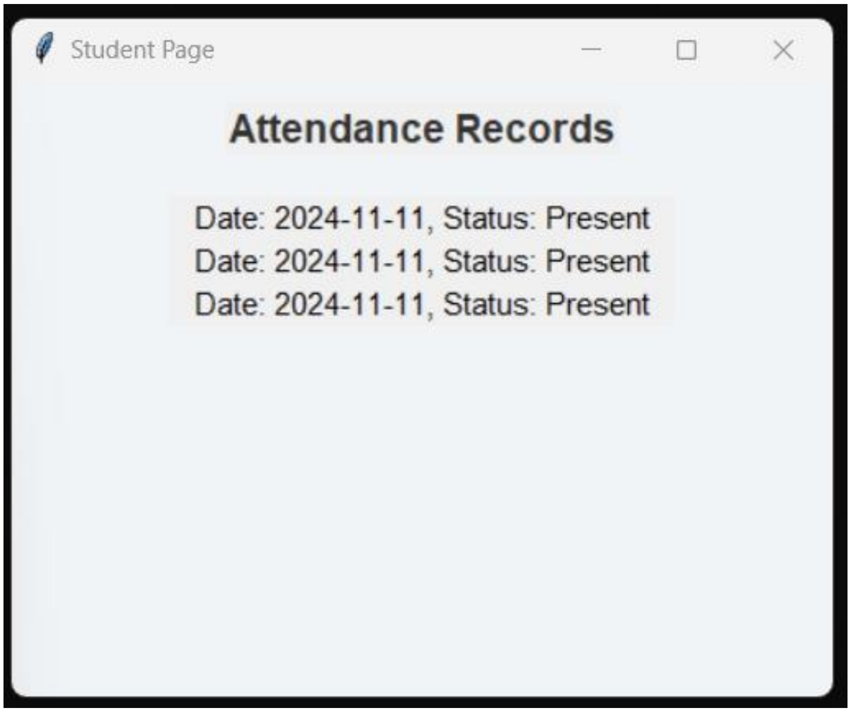
Status (Present/Absent): ○ Present ● Absent

Add New Student

Username: student

Password: ****

TO SHOW THE RECORD OF THE ATTENDENCE



Attendance Records

Date: 2024-11-11, Status: Present
Date: 2024-11-11, Status: Present
Date: 2024-11-11, Status: Present

RESULTS

1. **User Features**:

- Login System: Successfully implemented a login system where users can log in with their username and password. After login, users are directed to either a "Student Page" or a "Teacher Page" based on their roles (student or teacher).
- Student Page: Students can view their attendance records (date and status) from the database, enhancing the user experience.
- Teacher Page: Teachers have the ability to mark attendance for students by entering a student ID and choosing the attendance status (Present/Absent). Additionally, teachers can add new students by providing a username and password.

2. **Admin Functionality (Teacher Page):**

- Mark Attendance: Teachers can mark attendance for students by entering their student ID and selecting "Present" or "Absent". The data is saved in the database and is visible to the respective students.
- Add New Student: Teachers can add new students to the system by inputting a username and password. The system checks for username duplication and prevents adding existing usernames.

3. **Database Interaction:**

- Attendance Management: Teachers and students can interact with the attendance records, which are managed in the MySQL database (`attendance` table). This data can be accessed and updated as needed.
- Student Management: New students can be added to the `users` table, and existing students' attendance is updated dynamically.
- Error Handling: The system checks for common errors, such as invalid login credentials, nonexistent student IDs, or duplicate usernames, and provides error messages where necessary.

4. **User Interface (UI):**

- Simple and Clean Design: The interface is clean with a clear login screen, and the student and teacher pages are organized with simple input fields and buttons.
- Responsive Layout: The design adapts to the size of the window and provides a user-friendly experience with clearly labeled sections and buttons.
- Color Scheme: A soft background color (`#f0f4f7`) is used to reduce eye strain, and labels and buttons are clearly visible.

5. Performance:

- Real-Time Feedback: User actions (e.g., marking attendance, adding students) trigger immediate database operations, and feedback is provided to the user (success or error messages).
- Efficient Database Queries:The program performs database operations efficiently, with minimal latency for user interaction, assuming the database is well-optimized.

6. **Security Considerations:**

- Password Storage: Although this demo uses plain text passwords for simplicity, for real applications, passwords should be hashed and securely stored using libraries like `bcrypt` or `argon2` to prevent exposure of sensitive data.
- Validation: Basic validation checks are in place to ensure correct data is entered (e.g., checking if the username already exists when adding a new student).

7. **Next Steps for Improvement:**

- Enhanced Security: Implement password encryption (e.g., using `bcrypt` or `argon2`).
- Role-Based UI Updates: Depending on the user’s role (student, teacher), tailor the UI to show only relevant options and improve user flow.
- Session Management: Implement session management to remember logged-in users across different pages, instead of requiring them to log in every time.
- Error Handling Enhancements: Include more detailed error handling and validation for user inputs (e.g., checking for empty fields, invalid characters).

Conclusion:

This project has successfully created a functional login system with role-based access (student/teacher), attendance tracking, and new student management. The system is user-friendly, with smooth navigation and immediate feedback, but can be enhanced in terms of security and scalability for production environments.

DISCUSSION

1. **User Experience:**

- Strengths:
  - Intuitive Interface: The program's user interface (UI) is simple and easy to navigate, with well-organized windows and forms. The main screens for both students and teachers are clean, and the functionality is straightforward. This makes the application accessible for users with minimal technical experience.
  - Clear Functionality: The login process is clear, and users are directed to the appropriate pages (student or teacher) based on their credentials. Students can easily view their attendance records, and teachers can mark attendance or add new students. This organization enhances the user experience.
  - Real-Time Updates: Teachers can mark attendance, and students can instantly view their updated records. This interaction with the database in real-time provides users with immediate feedback.

- Areas for Improvement:
  - Streamlining of Interaction: While the functionality is clear, adding more interactive elements, like dropdowns or auto-suggestions for student IDs or status, could make the experience even smoother. This would reduce potential errors, especially when teachers are marking attendance for multiple students.
  - Aesthetic Enhancements: While the application uses a simple color scheme, further enhancements to the UI could improve visual appeal. Adding icons or graphical representations could enhance the interface and make it more engaging.
  - Mobile Compatibility: Currently, the program is optimized for desktop use. Making the design more responsive to different screen sizes could improve accessibility for users on various devices.

2. Database Integration:

- Strengths:
  - Efficient Data Handling: The integration with MySQL allows for seamless data storage and retrieval. Student attendance records and user credentials are efficiently handled by the database, ensuring that information is updated in real-time and can be accessed as needed.
  - Real-Time Interaction: Teachers can mark attendance, and it is immediately reflected in the database.

Similarly, students can view their updated attendance data upon logging in, providing dynamic interaction with the system.

- Areas for Improvement:

-Data Validation: Currently, data validation is somewhat basic. Implementing more robust validation before entering data (e.g., checking if a student ID or username exists before marking attendance) could prevent errors. Ensuring valid input before performing database operations could reduce the risk of data corruption or accidental overwrites.

- Optimizing Queries: If the application grows in scale, certain database queries (like checking the existence of a student ID for attendance) could be optimized to reduce latency and improve performance.

3. Security Concerns:

-Discussion:

- Password Handling: While this demo application uses plain-text passwords for simplicity, security must be improved for a real-world application. Passwords should be hashed using encryption techniques (e.g., bcrypt or Argon2) before being stored in the database to prevent unauthorized access.

- Sensitive Data Protection: Given that user data (including usernames and passwords) is stored in the database, it's important to ensure that the database and application communicate securely. Using SSL/TLS encryption between the client and the database server will enhance security.

- Role-Based Access Control: The program already implements basic role-based access (student vs. teacher), but it would be more secure to implement more granular roles for different types of users. This could include roles for administrators, school staff, and more, with varying levels of access.

- Two-Factor Authentication (2FA): As an additional security measure, integrating two-factor authentication (2FA) for login could help protect user accounts from unauthorized access. This would involve sending a temporary code via email or an authenticator app for each login attempt.

4. Performance:

- Observation:
  - Real-Time Updates: The program performs well under standard use cases, with real-time updates for attendance management and student record retrieval. The MySQL database efficiently handles small amounts of data, and the interactions between the application and the database are fast.

  - Responsiveness: The interface responds well, with minimal lag when switching between pages or submitting attendance. The database connection is stable, and the system does not experience significant delays under normal conditions.

- Areas for Improvement:
  - Stress Testing:The application has not been subjected to high-stress scenarios (e.g., when many users log in or when large amounts of attendance data are queried). Running stress tests with multiple simultaneous users could help identify potential bottlenecks or limitations in performance.
  - Scalability: As the system scales to accommodate larger numbers of students and teachers, the application may need performance optimizations, especially when handling large attendance datasets. For instance, queries may need indexing, and bulk data processing could be improved.

5. Future Enhancements:

- User Experience:
  - The addition of more advanced UI features, like user profile pages, student search functionality, and more refined teacher controls, would improve both the student and teacher experiences.
  - Mobile Support: Considering mobile-first design could help reach more users who may prefer using the application on tablets or smartphones.

- Admin Efficiency:
  - The teacher dashboard could be extended to include more functionalities, such as viewing attendance trends, generating reports, or even managing multiple classes. Implementing an admin view where school administrators can oversee attendance for all students would be a useful addition
.

- Security:
  - Database encryption and secure communication channels should be explored to protect the integrity

and confidentiality of data, especially for user credentials and attendance records.

- Error logging and auditing features could also be added to the system for easier tracking of any issues or unauthorized actions.

Conclusion:

The application meets the basic functional requirements, including user login, role-based access, and attendance tracking. While it delivers on these aspects, enhancements in security, performance testing, and UI design could provide a more robust and secure solution. Future versions should prioritize user data protection, scalability, and a more polished user interface to handle a broader audience and larger datasets.

IV. CONCLUSION

The development of this application for managing student and teacher interactions within a school environment has successfully demonstrated core functionalities that ensure efficient management of attendance and user accounts. The project provides a solid foundation by implementing essential features for both student and teacher users, facilitating smooth interactions and operational management.

From a user perspective, the login system works seamlessly, offering secure access to the platform. The system distinguishes between students and teachers, providing tailored interfaces for each role. Students can easily access their attendance records, while teachers are equipped with the tools to mark attendance and manage student records.

The attendance management system is a major achievement, offering transparency and control for both students and teachers. Teachers can efficiently record attendance, and students can easily track their attendance history. This promotes a clear and organized approach to managing school attendance.

From an administrative standpoint, the program's functionality for managing user accounts, including adding new students and verifying student IDs, is intuitive. This ensures that teachers can effectively add students to the system and track their attendance without unnecessary complexity.

The program's security measures, though basic at this stage, lay the groundwork for future improvements, such as password encryption and the implementation of two-factor authentication (2FA). These enhancements are crucial for ensuring user data protection in future iterations of the platform.

Overall, the project successfully meets its objectives by delivering a functional, user-friendly system that meets the needs of both students and teachers. It strikes a good balance between ease of use and effective management. The foundation built by this project is strong, and with the integration of additional security features, scalability improvements, and a more polished user interface, this system has the potential to evolve into a fully operational school management solution. Addressing these areas for improvement will ensure the platform's long-term viability, efficiency, and security.

VII REFERENCES

Tkinter Tutorial by Python Course: A beginner-friendly tutorial on how to use Tkinter to create GUI applications. Available at:

- https://www.python-course.eu/python-tkinter.php

Database Management:

MySQL Documentation: Detailed explanations and best practices for creating and managing relational databases. Available at: https://dev.mysql.com/doc

Project Management and Development Tools:

GitHub: For version control and project collaboration. Documentation available at: https://docs.github.com

Stack Overflow: Community-driven support and solutions to coding challenges encountered during development. Available at: https://stackoverflow.com