PM SHRI KENDRIYA VIDYALAYA NO.2 JAIPUR

तत् त्वं पूषन् अपावृणु
केन्द्रीय विद्यालय संगठन

<u>ACADEMIC YEAR : 2025-26</u>

<u>PROJECT REPORT ON</u>

"SCHOOL MANAGEMENT SYSTEM"

- **Name:** Piyus Saniya
- **Roll No:** 12324
- **Class:** XII – 'C'
- **Subject:** Computer Science
- **Sub Code:** 083
- **Project Guide:** Ms. Tarannum, PGT (CS)

<u>CERTIFICATE</u>

This is to certify that the project titled ***“SCHOOL MANAGEMENT SYSTEM”*** submitted by ***“PIYUS SANIYA”,***student of Class XII C, Roll No. ***12324,*** for the subject ***Computer Science (083),*** has been completed under my supervision.

This project is the original work of the student and full fills the requirements of the CBSE curriculum for the academic year 2025-26.

Sign. Of Internal Examiner

Sign. Of External Examiner

SIGN. OF PRINCIPAL

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I express my sincere gratitude to my teacher **“Ms. Tarannum Shaikh”** for their guidance and support throughout this project. I also thank our Principal **“Mr. Pradeep Kumar Tailor”** and my school for providing all necessary facilities.

I am grateful to my parents and friends for their constant support.

PIYUS SANIYA
Class XII - C
Roll No. 12324

DECLARATION

I hereby declare that the project titled ***“SCHOOL MANAGEMENT SYSTEM”*** is my original work and has not been submitted elsewhere. All information and data used in the project have been acknowledged properly.

Student Name: PIYUS SANIYA

Class: XII - C

Roll No: 12324

Signature: ______________

INTRODUCTION

The **School Management System** is a computer-based application developed using **Python** and **MySQL** that helps to manage and organize school-related information efficiently. This project aims to digitalize the traditional methods of record keeping in schools by storing all student and teacher data in a single, structured database.

The system allows easy management of **student records**, **teacher details**, **attendance**, **marks**, and **contact information**. Through this project, the tasks of adding, updating, and retrieving information become quick and error-free compared to manual registers.

This project also demonstrates how Python can be integrated with MySQL to create a fully functional, menu-driven database management system. By using simple commands and logical programming, the School Management System makes administrative work easier, reduces paper usage, and provides a clear example of how technology supports effective school management.

It not only strengthens the understanding of database connectivity and Python programming but also shows the practical application of these skills in real-world educational environment

OBJECTIVES OF THE PROJECT

The main objective of the **School Management System** is to simplify and automate the day-to-day administrative and academic tasks of a school. This project provides an efficient way to store, manage, and retrieve school data using Python and MySQL.

The specific objectives of this project are:

1. To design a computerized system that replaces the manual record-keeping process in schools.
2. To maintain accurate and up-to-date records of students and teachers, including attendance, marks, and personal details.
3. To demonstrate the use of **Python–MySQL connectivity** for managing databases effectively.
4. To make data retrieval, updating, and deletion operations faster and more convenient.
5. To reduce paperwork and human errors in handling school data.
6. To help students understand the importance of structured programming and database management systems in real-world scenarios.
7. To create a project that enhances logical thinking, coding skills, and the understanding of relational databases.

This project not only fulfills the requirements of the Class XII Computer Science curriculum but also helps in building problem-solving and database-handling skills among students.

HARDWARE AND SOFTWARE REQUIREMENTS

To successfully develop and execute the **School Management System**, the following hardware and software configurations were used:

"Hardware Requirements"

1. **Processor:** Intel Core i3 or above
2. **RAM:** Minimum 4 GB (8 GB recommended)
3. **Storage:** At least 100 MB of free disk space for project files and database
4. **Monitor:** Standard display unit
5. **Keyboard and Mouse:** For data input and navigation
6. **Operating System:** Windows 10 or above / Linux

"Software Requirements"

1. **Programming Language:** Python 3.8 or above
2. **Database:** MySQL Server 8.0
3. **Python Library:** `mysql.connector` (for database connectivity)
4. **Text Editor / IDE:** IDLE / VS Code / PyCharm
5. **Operating System Environment:** Windows or Ubuntu
6. **Command Line or Terminal:** To run Python programs and execute MySQL queries

This configuration ensures smooth functioning of the program, quick database transactions, and user-friendly interaction through the command-line interface.

SOURCE CODE

==

```
CREATE DATABASE school_db;

USE school_db;


CREATE TABLE teachers (

    teacher_id INT PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    designation VARCHAR(100) NOT NULL

);


CREATE TABLE teacher_attendance (

    id INT AUTO_INCREMENT PRIMARY KEY,

    teacher_id INT NOT NULL,

    att_date DATE NOT NULL,

    status ENUM('present','absent') NOT NULL,

    FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id) ON DELETE
CASCADE

);


CREATE TABLE students (

    admission_no INT PRIMARY KEY,

    roll_no INT NOT NULL,

    name VARCHAR(100) NOT NULL,

    class VARCHAR(20) NOT NULL,
    section CHAR(5),

    contact_no VARCHAR(20)

);


CREATE TABLE marks (

    id INT AUTO_INCREMENT PRIMARY KEY,

    admission_no INT NOT NULL,

    subject VARCHAR(100) NOT NULL,

    marks INT,

    FOREIGN KEY (admission_no) REFERENCES students(admission_no) ON DELETE
CASCADE

);


CREATE TABLE student_attendance (

    id INT AUTO_INCREMENT PRIMARY KEY,

    admission_no INT NOT NULL,

    att_date DATE NOT NULL,

    status ENUM('present','absent') NOT NULL,

    FOREIGN KEY (admission_no) REFERENCES students(admission_no) ON DELETE
CASCADE

);
```

```
  ==================================================================
import mysql.connector
import datetime
import sys

# Database Configuration
DB_CONFIG = {
    'host': 'localhost',
    'user': 'root',
    'password': 'root',  # <--- Change this to your actual password
    'database': 'school_db'
}

def get_connection():
    """Return a new DB connection."""
    try:
        con = mysql.connector.connect(**DB_CONFIG)
        return con
    except mysql.connector.Error as err:
        print(f"Error connecting to database: {err}")
        sys.exit(1)

def safe_input(prompt):
    """Handle KeyboardInterrupt cleanly."""
    try:
        return input(prompt)
    except KeyboardInterrupt:
        print("\nInterrupted. Returning to menu.")
        return ''

# ---------- Teacher Functions ----------

def add_teacher():
    con = get_connection()
    cur = con.cursor()
    try:
        tid = int(safe_input('Teacher ID (int): ').strip())
        name = safe_input('Name: ').strip()
        desig = safe_input('Designation: ').strip()
        cur.execute('INSERT INTO teachers (teacher_id, name, designation)
VALUES (%s,%s,%s)', (tid, name, desig))
        con.commit()
        print('-> Teacher added successfully.')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def view_teachers():
    con = get_connection()
    cur = con.cursor()
    try:
        cur.execute('SELECT teacher_id, name, designation FROM teachers ORDER
BY teacher_id')
        rows = cur.fetchall()
        if not rows:
            print('No teachers found.')
            return
        print('\n--- Teachers List ---')
        print(f"{'ID':<10} {'Name':<25} {'Designation':<20}")
        print("-" * 55)
        for r in rows:
            print(f"{r[0]:<10} {r[1]:<25} {r[2]:<20}")
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def update_teacher():
    con = get_connection()
    cur = con.cursor()
    try:
        tid = int(safe_input('Teacher ID to update: ').strip())
        field = safe_input('Field to update (name/designation):
').strip().lower()
        if field not in ('name','designation'):
            print('Invalid field.')
            return
        val = safe_input('New value: ').strip()
        cur.execute(f'UPDATE teachers SET {field}=%s WHERE teacher_id=%s', 
(val, tid))
        con.commit()
        print('-> Teacher updated.')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def delete_teacher():
    con = get_connection()
    cur = con.cursor()
    try:
        tid = int(safe_input('Teacher ID to delete: ').strip())
        confirm = safe_input('Type YES to confirm deletion: ')
        if confirm != 'YES':
            print('Canceled.')
            return
        cur.execute('DELETE FROM teachers WHERE teacher_id=%s', (tid,))
        con.commit()
        print('-> Deleted (if existed).')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def mark_teacher_attendance():
    con = get_connection()
    cur = con.cursor()
    try:
        tid = int(safe_input('Teacher ID: ').strip())
        date_str = safe_input('Date (YYYY-MM-DD) leave blank for today: 
').strip()
        if date_str == '':
            att_date = datetime.date.today()
        else:
            att_date = datetime.date.fromisoformat(date_str)
        status = safe_input('Status (present/absent): ').strip().lower()
        if status not in ('present','absent'):
            print('Invalid status.')
            return

        cur.execute('INSERT INTO teacher_attendance (teacher_id, att_date,
status) VALUES (%s,%s,%s)', (tid, att_date, status))
        con.commit()
        print('-> Attendance recorded.')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def view_teacher_attendance():
    con = get_connection()
    cur = con.cursor()
    try:
        tid = int(safe_input('Teacher ID: ').strip())
        cur.execute('SELECT att_date, status FROM teacher_attendance WHERE
teacher_id=%s ORDER BY att_date', (tid,))
        rows = cur.fetchall()
        if not rows:
            print('No attendance records.')
            return
        for r in rows:
            print(f'{r[0].isoformat()} \t {r[1]}')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

# ---------- Student Functions ----------

def add_student():
    con = get_connection()
    cur = con.cursor()
    try:
        adm = int(safe_input('Admission No (int): ').strip())
        roll = int(safe_input('Roll No (int): ').strip())
        name = safe_input('Name: ').strip()
        clas = safe_input('Class: ').strip()
        sec = safe_input('Section: ').strip()
        contact = safe_input('Contact No: ').strip()

        cur.execute('INSERT INTO students (admission_no, roll_no, name,
class, section, contact_no) VALUES (%s,%s,%s,%s,%s,%s)',
                    (adm, roll, name, clas, sec, contact))
        con.commit()
        print('-> Student added.')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def view_students():
    con = get_connection()
    cur = con.cursor()
    try:
        cur.execute('SELECT admission_no, roll_no, name, class, section,
contact_no FROM students ORDER BY class, section, roll_no')
        rows = cur.fetchall()
        if not rows:
            print('No students found.')
            return
        print('\n--- Student List ---')
        for s in rows:
            print(f'Adm: {s[0]} | Roll: {s[1]} | Name: {s[2]} | Class:
{s[3]}-{s[4]} | Ph: {s[5]}')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def update_student():
    con = get_connection()
    cur = con.cursor()
    try:
        adm = int(safe_input('Admission No to update: ').strip())
        field = safe_input('Field (roll_no/name/class/section/contact_no): 
').strip().lower()
        allowed = ('roll_no','name','class','section','contact_no')
        if field not in allowed:
            print('Invalid field.')
            return
        val = safe_input('New value: ').strip()
        if field == 'roll_no':
            val = int(val)
        cur.execute(f'UPDATE students SET {field}=%s WHERE admission_no=%s', 
(val, adm))
        con.commit()
        print('-> Student updated.')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def delete_student():
    con = get_connection()
    cur = con.cursor()
    try:
        adm = int(safe_input('Admission No to delete: ').strip())
        confirm = safe_input('Type YES to confirm deletion: ')
        if confirm != 'YES':
            print('Canceled.')
            return
        cur.execute('DELETE FROM students WHERE admission_no=%s', (adm,))
        con.commit()
        print('-> Deleted (if existed).')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

# ---------- Marks & Search Functions ----------

def add_mark():
    con = get_connection()
    cur = con.cursor()
    try:
        adm = int(safe_input('Admission No: ').strip())
        subject = safe_input('Subject: ').strip()
        marks_val = int(safe_input('Marks (int): ').strip())
        cur.execute('INSERT INTO marks (admission_no, subject, marks) VALUES
(%s,%s,%s)', (adm, subject, marks_val))
        con.commit()
        print('-> Mark added.')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def view_marks():
    con = get_connection()
    cur = con.cursor()
    try:
        adm = int(safe_input('Admission No: ').strip())
        cur.execute('SELECT subject, marks FROM marks WHERE admission_no=%s',
(adm,))
        rows = cur.fetchall()
        if not rows:
            print('No marks found.')
            return
        total = 0
        print(f'\n--- Marks for Adm No: {adm} ---')
        for r in rows:
            print(f'Subject: {r[0]} | Marks: {r[1]}')
            total += (r[1] or 0)
        avg = (total / len(rows)) if rows else 0
        print(f'Total: {total} | Average: {avg:.2f}')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

def search_student_by_name():
    con = get_connection()
    cur = con.cursor()
    try:
        name = safe_input('Enter name (partial ok): ').strip()
        cur.execute('SELECT admission_no, roll_no, name, class, section FROM 
students WHERE name LIKE %s', (f'%{name}%',))
        rows = cur.fetchall()
        if not rows:
            print('No matches.')
            return
        for s in rows:
            print(f'Adm:{s[0]} Roll:{s[1]} Name:{s[2]} Class:{s[3]}-{s[4]}')
    except Exception as e:
        print('Error:', e)
    finally:
        cur.close(); con.close()

# ---------- Menus ----------

def teacher_section_menu():
    while True:
        print('\n--- TEACHER SECTION ---')
        print('1. Add Teacher')
        print('2. View Teachers')
        print('3. Update Teacher')
        print('4. Delete Teacher')
        print('5. Mark Teacher Attendance')
        print('6. View Teacher Attendance')
        print('0. Back')
        ch = safe_input('Choice: ').strip()
        if ch == '1': add_teacher()
        elif ch == '2': view_teachers()
        elif ch == '3': update_teacher()
        elif ch == '4': delete_teacher()
        elif ch == '5': mark_teacher_attendance()
        elif ch == '6': view_teacher_attendance()
        elif ch == '0': break
        else: print('Invalid choice.')
```

```python
def student_section_menu():
    while True:
        print('\n--- STUDENT SECTION ---')
        print('1. Add Student')
        print('2. View Students')
        print('3. Update Student')
        print('4. Delete Student')
        print('5. Add Mark')
        print('6. View Marks')
        print('9. Search Student by Name')
        print('0. Back')
        ch = safe_input('Choice: ').strip()
        if ch == '1': add_student()
        elif ch == '2': view_students()
        elif ch == '3': update_student()
        elif ch == '4': delete_student()
        elif ch == '5': add_mark()
        elif ch == '6': view_marks()
        elif ch == '9': search_student_by_name()
        elif ch == '0': break
        else: print('Invalid choice.')

def main_menu():
    print('=============================================')
    print('      SCHOOL MANAGEMENT SYSTEM (v1.0)        ')
    print('=============================================')
    while True:
        print('\n=== MAIN MENU ===')
        print('1. Teacher Section')
        print('2. Student Section')
        print('0. Exit')
        ch = safe_input('Choice: ').strip()
        if ch == '1': teacher_section_menu()
        elif ch == '2': student_section_menu()
        elif ch == '0':
            print('Exiting System. Goodbye!')
            break
        else:
            print('Invalid choice.')
```

```
if __name__ == "__main__":
    main_menu()
 ======================================================================
```

OUTPUT

```
--------------------------------------------
       SCHOOL MANAGEMENT SYSTEM
--------------------------------------------
1. Teacher Section
2. Student Section
0. Exit
Enter choice: 2

--- STUDENT SECTION ---
1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Add Marks
6. View Marks
7. Mark Attendance
8. View Attendance
0. Back
Enter choice: 1

Admission No: 1001
Roll No: 12
Name: Riya Sharma
Class: 12
Section: C
Contact No: 9876543210
Student added successfully!
--------------------------------------------

Enter choice: 2
--- LIST OF STUDENTS ---
```

```
Adm: 1001   Roll: 12   Name: Riya Sharma   Class: 12   Sec: C   Contact:
9876543210


-------------------------------------------
Enter choice: 5
Admission No: 1001
Subject: Computer Science
Marks: 95
Marks added successfully!
-------------------------------------------


Enter choice: 6
Admission No: 1001
Subject: Computer Science   Marks: 95
Total: 95   Average: 95.00


-------------------------------------------
Enter choice: 7
Admission No: 1001
Date (YYYY-MM-DD) leave blank for today:
Status (present/absent): present
Attendance recorded successfully!


-------------------------------------------
Enter choice: 8
Admission No: 1001
2025-11-05   present


-------------------------------------------
Enter choice: 0
Returning to main menu...
```

```
--------------------------------------------
       SCHOOL MANAGEMENT SYSTEM
--------------------------------------------
1. Teacher Section
2. Student Section
0. Exit
Enter choice: 1

--- TEACHER SECTION ---
1. Add Teacher
2. View Teachers
3. Update Teacher
4. Delete Teacher
5. Mark Teacher Attendance
6. View Teacher Attendance
0. Back
Enter choice: 1

Teacher ID: 201
Name: Ms. Tarannum Shaikh
Designation: PGT – Computer Science
Teacher added successfully!

--------------------------------------------
Enter choice: 2
ID: 201   Name: Ms. Tarannum Shaikh   Designation: PGT – Computer Science
--------------------------------------------

Enter choice: 0
Returning to main menu...
Enter choice: 0
Goodbye!
```

CONCLUSION

The School Management System project successfully demonstrates how computer applications can simplify and automate the management of academic data. Using Python for programming and MySQL for database storage, this project shows the practical use of database connectivity in solving real-life problems related to data handling and record management.
The system efficiently maintains records of students and teachers, including details such as attendance, marks, and contact information. It eliminates the need for manual registers, making the process more accurate, organized, and time-efficient.

Through this project, I have gained a deeper understanding of how Python interacts with MySQL databases, how CRUD (Create, Read, Update, Delete) operations are performed, and how logical thinking helps in developing user-friendly systems.

This project has not only enhanced my programming and database skills but also given me confidence to design and implement more advanced applications in the future.

BIBLIOGRAPHY

While completing this project, I referred to various books, online resources, and class notes to understand the concepts of Python and MySQL. The following references were particularly helpful:

1. CBSE Computer Science (Class XII) Textbook, NCERT Publication
2. Sumita Arora, Computer Science with Python, Dhanpat Rai & Co.
3. Python Official Documentation – https://www.python.org/doc/
4. MySQL Official Documentation – https://dev.mysql.com/doc/
5. Classroom Lectures and Practical Notebook by Mrs. Tarannum Shaikh (PGT-CS)
