CONTENTS

♣ Introduction ♣

♣ Objective:
The Student Attendance System is a GUI-based application developed using Python, MySQL, and IPyWidgets. The main goal of this project is to digitally manage student attendance records, making the process efficient, accurate, and easy to access. It will include features like student record management (Add, View, Update, Delete).

♣ Features of the Project:
✅ **Add Students** – Register new students in the system.
✅ **Mark Attendance** – Record student attendance as "Present" or "Absent".
✅ **View Attendance Records** – Retrieve attendance history of students.

♣ Technologies Used:
Python – For backend logic and database connectivity.
- MySQL-To store student details and attendance records.
- iPyWidgets– To create an interactive GUI in Jupyter Notebook.

♣ Scope of the Project:

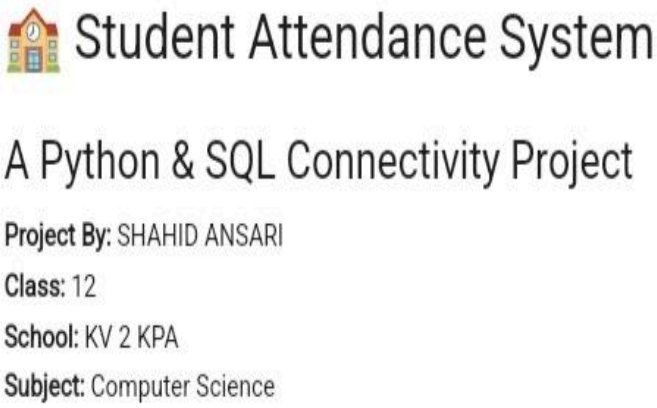
- Helps teachers track attendance digitally.
- Reduces errors and makes data retrieval easier.
- Can be expanded to include report generation and analytics.

♣ Title Page ♣

Code:-

```
from IPython.display import display, Markdown
title = "# 🏫 Student Attendance System\n## A Python & SQL Connectivity Project"
student_details = "**Project By:** Shahid Ansari \n**Class:** 12  \n**School:** KV 2 KPAI  \n**Subject:** Computer Science"

display(Markdown(title))
display(Markdown(student_details))
```

[44]:
```
from IPython.display import display, Markdown

title = "# 🏫 Student Attendance System\n## A Python & SQL Connectivity Project"
student_details = "**Project By:** SHAHID ANSARI  \n**Class:** 12  \n**School:** KV 2 KPA  \n**Subject:** Compute

display(Markdown(title))
display(Markdown(student_details))
```

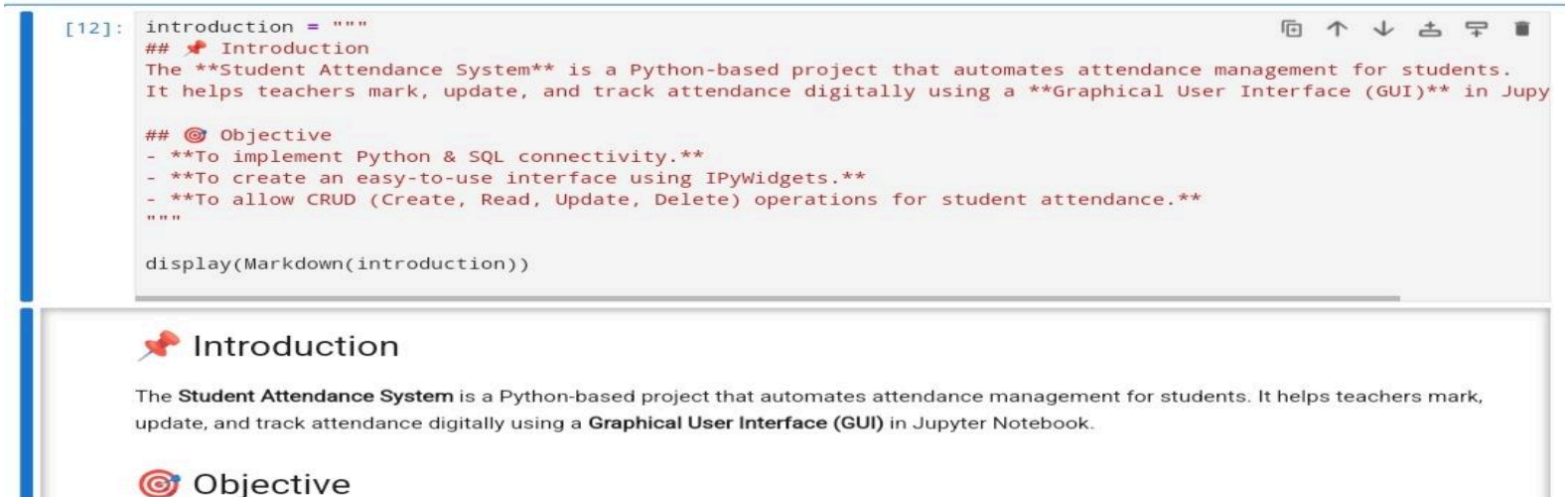


✅ Displays project details in a structured format.

♣ Introduction & Objective ♣

Code:-

```
introduction = """
## 📌 Introduction
The **Student Attendance System** is a Python-based project that automates attendance management for students.
It helps teachers mark, update, and track attendance digitally using a **Graphical User Interface (GUI)** in Jupyter Notebook.

## 🎯 Objective
- **To implement Python & SQL connectivity.**
- **To create an easy-to-use interface using IPyWidgets.**
- **To allow CRUD (Create, Read, Update, Delete) operations for student attendance.**
"""

display(Markdown(introduction))
```



♣ Installing Required Libraries and setting up mysql database ♣

Code for installing libraries:-

>>> pip install mysql-connector-python ipywidgets

✅ Ensures all dependencies are installed.

Setting up mysql database:-

>>>

```
import mysql.connector

conn = mysql.connector.connect(host="localhost",
user="root", password="password")
cursor = conn.cursor()

cursor.execute("CREATE DATABASE IF NOT EXISTS
student_attendance_db")
conn.database = "student_attendance_db"

cursor.execute("""
    CREATE TABLE IF NOT EXISTS students (
        roll_number INT PRIMARY KEY,
        name VARCHAR(100) NOT NULL,
        class VARCHAR(20) NOT NULL
    )
""")
```

```
cursor.execute("""
    CREATE TABLE IF NOT EXISTS attendance (
        id INT AUTO_INCREMENT PRIMARY KEY,
        roll_number INT,
        date DATE NOT NULL,
        status ENUM('Present', 'Absent') NOT NULL,
        FOREIGN KEY (roll_number) REFERENCES
students(roll_number) ON DELETE CASCADE
    )
""")

display(Markdown("✅ **Database and Tables Created
Successfully!**"))

cursor.close()
conn.close()
```

jupyter stud_atendnce_sys Last Checkpoint: 43 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[17]:
```
import mysql.connector

conn = mysql.connector.connect(host="127.0.0.1", user="root", password="shahid", database="student_attendance_db"
cursor = conn.cursor()

display(Markdown("✅ **Database Connection Successful!**"))
```

✅ **Database Connection Successful!**

[15]:
```
import mysql.connector

conn = mysql.connector.connect(host="127.0.0.1", user="root", password="shahid")
cursor = conn.cursor()

cursor.execute("CREATE DATABASE IF NOT EXISTS student_attendance_db")
conn.database = "student_attendance_db"

cursor.execute("""
    CREATE TABLE IF NOT EXISTS students (
        roll_number INT PRIMARY KEY,
        name VARCHAR(100) NOT NULL,
        class VARCHAR(20) NOT NULL
    )
""")

cursor.execute("""
    CREATE TABLE IF NOT EXISTS attendance (
        id INT AUTO_INCREMENT PRIMARY KEY,
        roll_number INT,
        date DATE NOT NULL,
        status ENUM('Present', 'Absent') NOT NULL,
        FOREIGN KEY (roll_number) REFERENCES students(roll_number) ON DELETE CASCADE
    )
""")

display(Markdown("✅ **Database and Tables Created Successfully!**"))

cursor.close()
conn.close()
```

✅ **Database and Tables Created Successfully!**

♣ Adding a student ♣

Code:-

```
add_student_output = widgets.Output()

def show_add_student_page():
    with add_student_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll
Number")
        name_input = widgets.Text(placeholder="Enter
Student Name")
        class_input = widgets.Text(placeholder="Enter
Class")
        submit_button = widgets.Button(description="Add
Student")

        def add_student(b):
            query = "INSERT INTO students (roll_number,
name, class) VALUES (%s, %s, %s)"
            cursor.execute(query, (roll_input.value,
name_input.value, class_input.value))
            conn.commit()
            with add_student_output:
```

```
            clear_output()
            display(widgets.Label("✅ Student Added
Successfully!"))

        submit_button.on_click(add_student)
        display(roll_input, name_input, class_input,
submit_button)

display(add_student_output)
```

[19]:
```
add_student_output = widgets.Output()

def show_add_student_page():
    with add_student_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll Number")
        name_input = widgets.Text(placeholder="Enter Student Name")
        class_input = widgets.Text(placeholder="Enter Class")
        submit_button = widgets.Button(description="Add Student")

        def add_student(b):
            query = "INSERT INTO students (roll_number, name, class) VALUES (%s, %s, %s)"
            cursor.execute(query, (roll_input.value, name_input.value, class_input.value))
            conn.commit()
            with add_student_output:
                clear_output()
                display(widgets.Label("✅ Student Added Successfully!"))

        submit_button.on_click(add_student)
        display(roll_input, name_input, class_input, submit_button)

display(add_student_output)
```

♣ Marking Attendance ♣

Code:-

```
mark_attendance_output = widgets.Output()

def show_mark_attendance_page():
    with mark_attendance_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll
Number")
        status_dropdown =
widgets.Dropdown(options=["Present", "Absent"],
description="Status:")
        submit_button = widgets.Button(description="Mark
Attendance")

        def mark_attendance(b):
            query = "INSERT INTO attendance (roll_number,
date, status) VALUES (%s, CURDATE(), %s)"
            cursor.execute(query, (roll_input.value,
status_dropdown.value))
            conn.commit()
            with mark_attendance_output:
                clear_output()
```

display(widgets.Label("✅ Attendance Marked Successfully!"))

submit_button.on_click(mark_attendance)
display(roll_input, status_dropdown, submit_button)

display(mark_attendance_output)

[33]:
```
mark_attendance_output = widgets.Output()

def show_mark_attendance_page():
    with mark_attendance_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll Number")
        status_dropdown = widgets.Dropdown(options=["Present", "Absent"], description="Status:")
        submit_button = widgets.Button(description="Mark Attendance")

        def mark_attendance(b):
            query = "INSERT INTO attendance (roll_number, date, status) VALUES (%s, CURDATE(), %s)"
            cursor.execute(query, (roll_input.value, status_dropdown.value))
            conn.commit()
            with mark_attendance_output:
                clear_output()
                display(widgets.Label("✅ Attendance Marked Successfully!"))

        submit_button.on_click(mark_attendance)
        display(roll_input, status_dropdown, submit_button)

display(mark_attendance_output)
```

✅ Marks attendance for the student.

♣ Updating student details ♣

Code:-

```
update_student_output = widgets.Output()

def show_update_student_page():
    with update_student_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll
Number")
        name_input = widgets.Text(placeholder="Enter New
Name")
        class_input = widgets.Text(placeholder="Enter New
Class")
        submit_button = widgets.Button(description="Update
Student")

        def update_student(b):
            query = "UPDATE students SET name=%s,
class=%s WHERE roll_number=%s"
            cursor.execute(query, (name_input.value,
class_input.value, roll_input.value))
            conn.commit()
            with update_student_output:
```

clear_output()
display(widgets.Label("✅ Student Updated Successfully!"))

submit_button.on_click(update_student)
display(roll_input, name_input, class_input, submit_button)

display(update_student_output)

[34]:
```
update_student_output = widgets.Output()

def show_update_student_page():
    with update_student_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll Number")
        name_input = widgets.Text(placeholder="Enter New Name")
        class_input = widgets.Text(placeholder="Enter New Class")
        submit_button = widgets.Button(description="Update Student")

        def update_student(b):
            query = "UPDATE students SET name=%s, class=%s WHERE roll_number=%s"
            cursor.execute(query, (name_input.value, class_input.value, roll_input.value))
            conn.commit()
            with update_student_output:
                clear_output()
                display(widgets.Label("✅ Student Updated Successfully!"))

        submit_button.on_click(update_student)
        display(roll_input, name_input, class_input, submit_button)

display(update_student_output)
```

✅ Student Updated Successfully!

✅ Allows modifying student details.

♣ Updating Attendance ♣

Code:-

```
update_attendance_output = widgets.Output()

def show_update_attendance_page():
    with update_attendance_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll
Number")
        date_input = widgets.Text(placeholder="Enter Date
(YYYY-MM-DD)")
        status_dropdown =
widgets.Dropdown(options=["Present", "Absent"],
description="New Status:")
        submit_button = widgets.Button(description="Update
Attendance")

        def update_attendance(b):
            query = "UPDATE attendance SET status=%s
WHERE roll_number=%s AND date=%s"
            cursor.execute(query, (status_dropdown.value,
roll_input.value, date_input.value))
            conn.commit()
```

```
            with update_attendance_output:
                clear_output()
                display(widgets.Label("✅ Attendance Updated
Successfully!"))

        submit_button.on_click(update_attendance)
        display(roll_input, date_input, status_dropdown,
submit_button)

display(update_attendance_output)
```

[40]:
```
update_attendance_output = widgets.Output()

def show_update_attendance_page():
    with update_attendance_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll Number")
        date_input = widgets.Text(placeholder="Enter Date (YYYY-MM-DD)")
        status_dropdown = widgets.Dropdown(options=["Present", "Absent"], description="New Status:")
        submit_button = widgets.Button(description="Update Attendance")

        def update_attendance(b):
            query = "UPDATE attendance SET status=%s WHERE roll_number=%s AND date=%s"
            cursor.execute(query, (status_dropdown.value, roll_input.value, date_input.value))
            conn.commit()
            with update_attendance_output:
                clear_output()
                display(widgets.Label("✅ Attendance Updated Successfully!"))

        submit_button.on_click(update_attendance)
        display(roll_input, date_input, status_dropdown, submit_button)

display(update_attendance_output)
```

✅ Attendance Updated Successfully!

✅ Updates attendance records of the student for a particular date.

♣ Deleting a student ♣

Code:-

```
delete_student_output = widgets.Output()

def show_delete_student_page():
    with delete_student_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll
Number to Delete")
        submit_button = widgets.Button(description="Delete
Student")

        def delete_student(b):
            query = "DELETE FROM students WHERE
roll_number=%s"
            cursor.execute(query, (roll_input.value,))
            conn.commit()
            with delete_student_output:
                clear_output()
                display(widgets.Label("✅ Student Deleted
Successfully!"))

        submit_button.on_click(delete_student)
```

display(roll_input, submit_button)

display(delete_student_output)

[36]:
```
delete_student_output = widgets.Output()

def show_delete_student_page():
    with delete_student_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll Number to Delete")
        submit_button = widgets.Button(description="Delete Student")

        def delete_student(b):
            query = "DELETE FROM students WHERE roll_number=%s"
            cursor.execute(query, (roll_input.value,))
            conn.commit()
            with delete_student_output:
                clear_output()
                display(widgets.Label("✅ Student Deleted Successfully!"))

        submit_button.on_click(delete_student)
        display(roll_input, submit_button)

display(delete_student_output)
```

✅ Deletes a student from the database by entry no.

♣ Deleting attendance ♣

Code:-

```
delete_attendance_output = widgets.Output()

def show_delete_attendance_page():
    with delete_attendance_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll
Number")
        date_input = widgets.Text(placeholder="Enter Date
(YYYY-MM-DD)")
        submit_button = widgets.Button(description="Delete
Attendance")

        def delete_attendance(b):
            query = "DELETE FROM attendance WHERE
roll_number=%s AND date=%s"
            cursor.execute(query, (roll_input.value,
date_input.value))
            conn.commit()
            with delete_attendance_output:
                clear_output()
```

display(widgets.Label("✅ Attendance Deleted Successfully!"))

submit_button.on_click(delete_attendance)
display(roll_input, date_input, submit_button)

display(delete_attendance_output)

[42]:
```
delete_attendance_output = widgets.Output()

def show_delete_attendance_page():
    with delete_attendance_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll Number")
        date_input = widgets.Text(placeholder="Enter Date (YYYY-MM-DD)")
        submit_button = widgets.Button(description="Delete Attendance")

        def delete_attendance(b):
            # Check if the attendance record exists
            check_query = "SELECT * FROM attendance WHERE roll_number=%s AND date=%s"
            cursor.execute(check_query, (roll_input.value, date_input.value))
            record = cursor.fetchone()

            if record:
                # If record exists, proceed with deletion
                delete_query = "DELETE FROM attendance WHERE roll_number=%s AND date=%s"
                cursor.execute(delete_query, (roll_input.value, date_input.value))
                conn.commit()
                with delete_attendance_output:
                    clear_output()
                    display(widgets.Label("✅ Attendance Deleted Successfully!"))
            else:
                # If no matching record found, show error message
                with delete_attendance_output:
                    clear_output()
                    display(widgets.Label("❌ No matching attendance record found!"))

        submit_button.on_click(delete_attendance)
        display(roll_input, date_input, submit_button)

display(delete_attendance_output)
```

✅ Deletes attendance records of a particular student.

Viewing Attendance

Code:-

```
view_attendance_output = widgets.Output()

def show_view_attendance_page():
    with view_attendance_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll
Number to View Attendance")
        submit_button = widgets.Button(description="View
Attendance")

        def view_attendance(b):
            query = "SELECT date, status FROM attendance
WHERE roll_number=%s"
            cursor.execute(query, (roll_input.value,))
            records = cursor.fetchall()

            with view_attendance_output:
                clear_output()
                if records:
                    for record in records:
```

```
                    display(widgets.Label(f"📅 Date:
{record[0]}, Status: {record[1]}"))
            else:
                display(widgets.Label("❌ No attendance
records found."))

    submit_button.on_click(view_attendance)
    display(roll_input, submit_button)

display(view_attendance_output)
```

[25]:
```
view_attendance_output = widgets.Output()

def show_view_attendance_page():
    with view_attendance_output:
        clear_output()

        roll_input = widgets.IntText(placeholder="Enter Roll Number to View Attendance")
        submit_button = widgets.Button(description="View Attendance")

        def view_attendance(b):
            query = "SELECT date, status FROM attendance WHERE roll_number=%s"
            cursor.execute(query, (roll_input.value,))
            records = cursor.fetchall()

            with view_attendance_output:
                clear_output()
                if records:
                    for record in records:
                        display(widgets.Label(f"📅 Date: {record[0]}, Status: {record[1]}"))
                else:

                    display(widgets.Label("❌ No attendance records found."))

        submit_button.on_click(view_attendance)
        display(roll_input, submit_button)

display(view_attendance_output)
```

📅 Date: 2025-02-04, Status: Present

✅ By giving input the entry no. of a student you can view the record of that student for all the dates in the past and whether he was absent or not on that day.

Creating the User Interface (Dashboard)

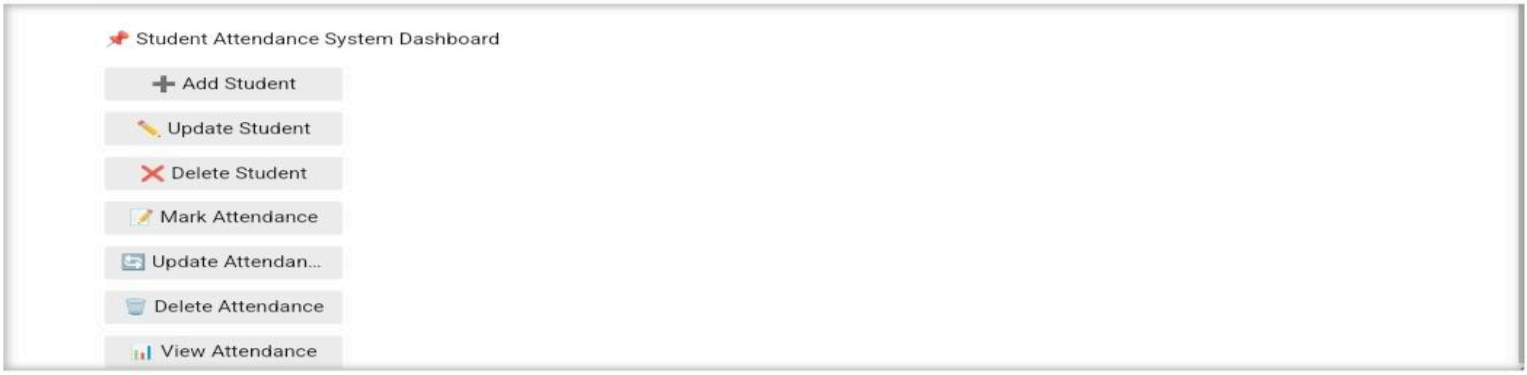
Code:-

```
import ipywidgets as widgets
from IPython.display import display, clear_output

dashboard_output = widgets.Output()

def show_dashboard():
    with dashboard_output:
        clear_output()

        display(widgets.Label("📌 Student Attendance
System Dashboard"))

        # Buttons
        buttons = [
            ("➕ Add Student", show_add_student_page),
            ("✏️ Update Student",
show_update_student_page),
            ("❌ Delete Student", show_delete_student_page),
            ("📝 Mark Attendance",
show_mark_attendance_page),
            ("🔄 Update Attendance",
show_update_attendance_page),
```

```
        ("🗑️ Delete Attendance",
show_delete_attendance_page),
        ("📊 View Attendance",
show_view_attendance_page),
    ]

    for text, function in buttons:
        btn = widgets.Button(description=text)
        btn.on_click(lambda b, f=function: f())
        display(btn)

show_dashboard()
display(dashboard_output)
```

[43]:

```
import ipywidgets as widgets
from IPython.display import display, clear_output

dashboard_output = widgets.Output()

def show_dashboard():
    with dashboard_output:
        clear_output()

        display(widgets.Label("📌 Student Attendance System Dashboard"))

        # Buttons
        buttons = [
            ("➕ Add Student", show_add_student_page),
            ("✏️ Update Student", show_update_student_page),
            ("❌ Delete Student", show_delete_student_page),
            ("📝 Mark Attendance", show_mark_attendance_page),
            ("🔄 Update Attendance", show_update_attendance_page),
            ("🗑️ Delete Attendance", show_delete_attendance_page),
            ("📊 View Attendance", show_view_attendance_page),
        ]

        for text, function in buttons:
            btn = widgets.Button(description=text)
            btn.on_click(lambda b, f=function: f())
            display(btn)

show_dashboard()
display(dashboard_output)
```



♣ Conclusion ♣

The Student Attendance System is a user-friendly application that allows teachers and administrators to efficiently manage student records and track attendance.

🔑 1. **Key Features Implemented:**
✅ Add, Update, Delete Student Records
✅ Mark Attendance & View Attendance Records
✅ Validate Attendance Deletion to Prevent Errors
✅ Store & Retrieve Data using MySQL & Python Connectivity

This project eliminates manual record-keeping and reduces errors in attendance tracking. It demonstrates the power of Python and MySQL integration in real-world applications.

🚀 **2. Future Enhancements**
In the future, this system can be enhanced with:

- 🔹 Graphical Reports – Generate attendance reports with charts & graphs using Matplotlib.
- 🔹 Login System – Add user authentication to restrict access to teachers/admins.
- 🔹 Face Recognition Attendance – Use OpenCV for biometric attendance marking.

🔹 Export Data to Excel – Allow attendance reports to be downloaded as Excel/PDF.

🔹 Mobile App Integration – Create a mobile-friendly interface for attendance tracking.

🎯 3. Final Thoughts

With the successful implementation of the Student Attendance System, this project showcases how Python and MySQL can work together to create efficient real-world applications.

♣ Database view ♣

```
| school                 |
| student_attendance_db  |
| sys                    |
| test                   |
+------------------------+
8 rows in set (0.012 sec)

        [(none)]> use student_attendance_db;
Reading table information for completion of table
 and column names
You can turn off this feature to get a quicker st
artup with -A

Database changed
        [student_attendance_db]> show tables
    -> ;
+-----------------------------------+
| Tables_in_student_attendance_db   |
+-----------------------------------+
| attendance                        |
| students                          |
+-----------------------------------+
2 rows in set (0.001 sec)

        [student_attendance_db]> select * from at
tendance;
+----+-------------+------------+--------+
| id | roll_number | date       | status |
+----+-------------+------------+--------+
|  3 |           3 | 2025-02-04 | Absent |
|  4 |          10 | 2025-02-04 | Absent |
|  5 |          12 | 2025-02-04 | Absent |
|  7 |           2 | 2025-02-04 | Absent |
+----+-------------+------------+--------+
4 rows in set (0.001 sec)

        [student_attendance_db]> select * from st
udents;
+-------------+---------+-------+
| roll_number | name    | class |
+-------------+---------+-------+
|           1 | Shahid  | 10    |
|           2 | Shabnam | 12    |
|           3 | Shabnam | 11    |
|          10 | Sohel   | 10    |
|          12 | Shahid  | 15    |
+-------------+---------+-------+
5 rows in set (0.002 sec)
```

♣ References ♣

During the project development, we used the following resources:

♠ Python Official Documentation –https://docs.python.org/

♠ MySQL Documentation – https://dev.mysql.com/doc/

♠ IPyWidgets Guide – https://ipywidgets.readthedocs.io/en/latest/

These references helped in understanding Python-SQL integration, database management, and GUI development using IPyWidgets.