

**STUDENTS ATTENDANCE MANAGER**

**NAME:**

**DEPARTMENT:**

**YEAR:**

**REGISTRATION NUMBER:**

## **Student Attendance Manager**

### **AIM:**

### To build a dynamic and interactive web-based attendance management system that allows users to add, manage, and analyze student attendance records efficiently.

### **Tech Stack:**

#### **Frontend:**

Streamlit – For interactive UI and easy deployment

#### **Backend:**

MySQL – For storing student and attendance data

Python (MySQL Connector) – For database interactions

Pandas – For data summarization and analysis

### **Application Workflow & Algorithm:**

#### **1.Initialize Database**

#### When the app starts, it connects to MySQL and:

#### Creates a database student\_db if not exists.

#### Creates two tables:

#### students: To store student names and roll numbers.

#### attendance: To store attendance records linked by student\_id.

#### **Sidebar Menu Options**

#### Users can choose from:

* Add Student
* View Students
* Delete Student
* Mark Attendance
* View Attendance

#### **3. Add Student**

Input: Student name.

Auto-generates next roll number

Inserts new student into students table.

#### **4. View Students**

* Displays a table of all registered students from the students table.

#### **5. Delete Student**

* Select a student by name.
* Confirms deletion.
* Removes student from students and cascades delete in attendance.

#### **6. Mark Attendance**

* Select a date (defaults to today).
* For each student, choose status: Present, Absent, or Leave.
* Inserts or updates record in the attendance table.

#### **7. View Attendance**

Three viewing options:

**All Records**

Shows a complete attendance log (roll no, name, date, status).

**Individual Student**

Filters and displays attendance for a selected student.

**Low Attendance Filter**

Filters students with attendance percentage below a user-defined threshold (slider).

#### **8. Attendance Summary Calculation**

Uses SQL and Pandas to compute:

* Total days
* Present days
* Attendance percentage

### **Program:**

import streamlit as st

import mysql.connector

from datetime import date

import pandas as pd

# ------------------- PAGE SETUP -------------------

st.set\_page\_config(page\_title="Student Attendance Manager", layout="centered")

# ------------------- INITIALIZE DATABASE & TABLES -------------------

def init\_database():

conn = mysql.connector.connect(host="localhost", user="root", password="12345678")

cursor = conn.cursor()

cursor.execute("CREATE DATABASE IF NOT EXISTS student\_db")

cursor.close()

conn.close()

conn = mysql.connector.connect(host="localhost", user="root", password="12345678", database="student\_db")

cursor = conn.cursor()

# Students table

cursor.execute("""

CREATE TABLE IF NOT EXISTS students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

roll\_no INT UNIQUE NOT NULL,

name VARCHAR(100) NOT NULL

)

""")

# Attendance table

cursor.execute("""

CREATE TABLE IF NOT EXISTS attendance (

id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT,

date DATE,

status ENUM('Present', 'Absent', 'Leave'),

UNIQUE(student\_id, date),

FOREIGN KEY (student\_id) REFERENCES students(student\_id) ON DELETE CASCADE

)

""")

conn.commit()

cursor.close()

conn.close()

init\_database()

# ------------------- MYSQL CONNECTION -------------------

def get\_connection():

return mysql.connector.connect(

host="localhost",

user="root",

password="12345678",

database="student\_db"

)

# ------------------- SIDEBAR MENU -------------------

menu = st.sidebar.selectbox("Menu", [

"➕ Add Student",

"📋 View Students",

"🗑️ Delete Student",

"✅ Mark Attendance",

"📊 View Attendance"

])

# ------------------- ADD STUDENT -------------------

if menu == "➕ Add Student":

st.title("Add a New Student")

name = st.text\_input("Enter Student Name")

if st.button("Add Student"):

if name.strip():

conn = get\_connection()

cursor = conn.cursor()

cursor.execute("SELECT IFNULL(MAX(roll\_no), 0) + 1 FROM students")

next\_roll = cursor.fetchone()[0]

try:

cursor.execute("INSERT INTO students (roll\_no, name) VALUES (%s, %s)", (next\_roll, name))

conn.commit()

st.success(f"Student '{name}' added successfully with Roll No {next\_roll}")

except mysql.connector.Error as err:

st.error(f"Error: {err}")

finally:

cursor.close()

conn.close()

else:

st.warning("Please enter a valid name.")

# ------------------- VIEW STUDENTS -------------------

elif menu == "📋 View Students":

st.title("List of Students")

conn = get\_connection()

cursor = conn.cursor()

cursor.execute("SELECT roll\_no, name FROM students ORDER BY roll\_no")

students = cursor.fetchall()

cursor.close()

conn.close()

if students:

st.table(students)

else:

st.info("No students found.")

# ------------------- DELETE STUDENT -------------------

elif menu == "🗑️ Delete Student":

st.title("Delete a Student")

conn = get\_connection()

cursor = conn.cursor()

cursor.execute("SELECT student\_id, name FROM students ORDER BY name")

students = cursor.fetchall()

if students:

student\_dict = {name: sid for sid, name in students}

selected\_name = st.selectbox("Select student to delete", list(student\_dict.keys()))

confirm = st.radio("Are you sure?", ["No", "Yes"])

if st.button("Delete") and confirm == "Yes":

sid = student\_dict[selected\_name]

cursor.execute("DELETE FROM students WHERE student\_id=%s", (sid,))

conn.commit()

st.success(f"Student '{selected\_name}' has been deleted.")

else:

st.info("No students to delete.")

cursor.close()

conn.close()

# ------------------- MARK ATTENDANCE -------------------

elif menu == "✅ Mark Attendance":

st.title("Mark Attendance")

selected\_date = st.date\_input("Select Date to Mark Attendance", date.today())

conn = get\_connection()

cursor = conn.cursor()

cursor.execute("SELECT student\_id, name FROM students ORDER BY name")

students = cursor.fetchall()

if students:

for sid, name in students:

status = st.radio(f"{name}", ["Present", "Absent", "Leave"], key=f"{sid}\_{selected\_date}")

if st.button(f"Save for {name}", key=f"save\_{sid}\_{selected\_date}"):

cursor.execute("""

INSERT INTO attendance (student\_id, date, status)

VALUES (%s, %s, %s)

ON DUPLICATE KEY UPDATE status=%s

""", (sid, selected\_date, status, status))

conn.commit()

st.success(f"{name}'s attendance for {selected\_date} marked as {status}")

else:

st.info("No students found.")

cursor.close()

conn.close()

# ------------------- VIEW ATTENDANCE -------------------

elif menu == "📊 View Attendance":

st.title("Attendance Records")

conn = get\_connection()

# Full attendance log

df = pd.read\_sql("""

SELECT s.student\_id, s.roll\_no AS RollNo, s.name AS Student, a.date AS Date, a.status AS Status

FROM attendance a

JOIN students s ON s.student\_id = a.student\_id

ORDER BY a.date DESC, s.roll\_no ASC

""", conn)

# Summary table

df\_summary = pd.read\_sql("""

SELECT s.roll\_no AS RollNo, s.name AS Student,

COUNT(\*) AS TotalDays,

SUM(CASE WHEN a.status = 'Present' THEN 1 ELSE 0 END) AS PresentDays,

ROUND(SUM(CASE WHEN a.status = 'Present' THEN 1 ELSE 0 END) \* 100 / COUNT(\*), 2) AS AttendancePercent

FROM attendance a

JOIN students s ON s.student\_id = a.student\_id

GROUP BY s.student\_id, s.roll\_no, s.name

ORDER BY s.roll\_no

""", conn)

conn.close()

# UI for filtering

view\_mode = st.radio("View Options", ["📄 All Records", "👤 Individual Student", "⚠️ Low Attendance Filter"])

if view\_mode == "📄 All Records":

if not df.empty:

st.subheader("Full Attendance Log")

st.dataframe(df)

else:

st.info("No attendance records found.")

elif view\_mode == "👤 Individual Student":

st.subheader("Select Student")

student\_names = df['Student'].unique().tolist()

selected\_student = st.selectbox("Choose a student", student\_names)

filtered\_df = df[df['Student'] == selected\_student]

if not filtered\_df.empty:

st.dataframe(filtered\_df)

else:

st.warning("No records for selected student.")

elif view\_mode == "⚠️ Low Attendance Filter":

st.subheader("Filter Students by Attendance Percentage")

threshold = st.slider("Show students with attendance below (%)", 0, 100, 60)

filtered\_summary = df\_summary[df\_summary['AttendancePercent'] < threshold]

if not filtered\_summary.empty:

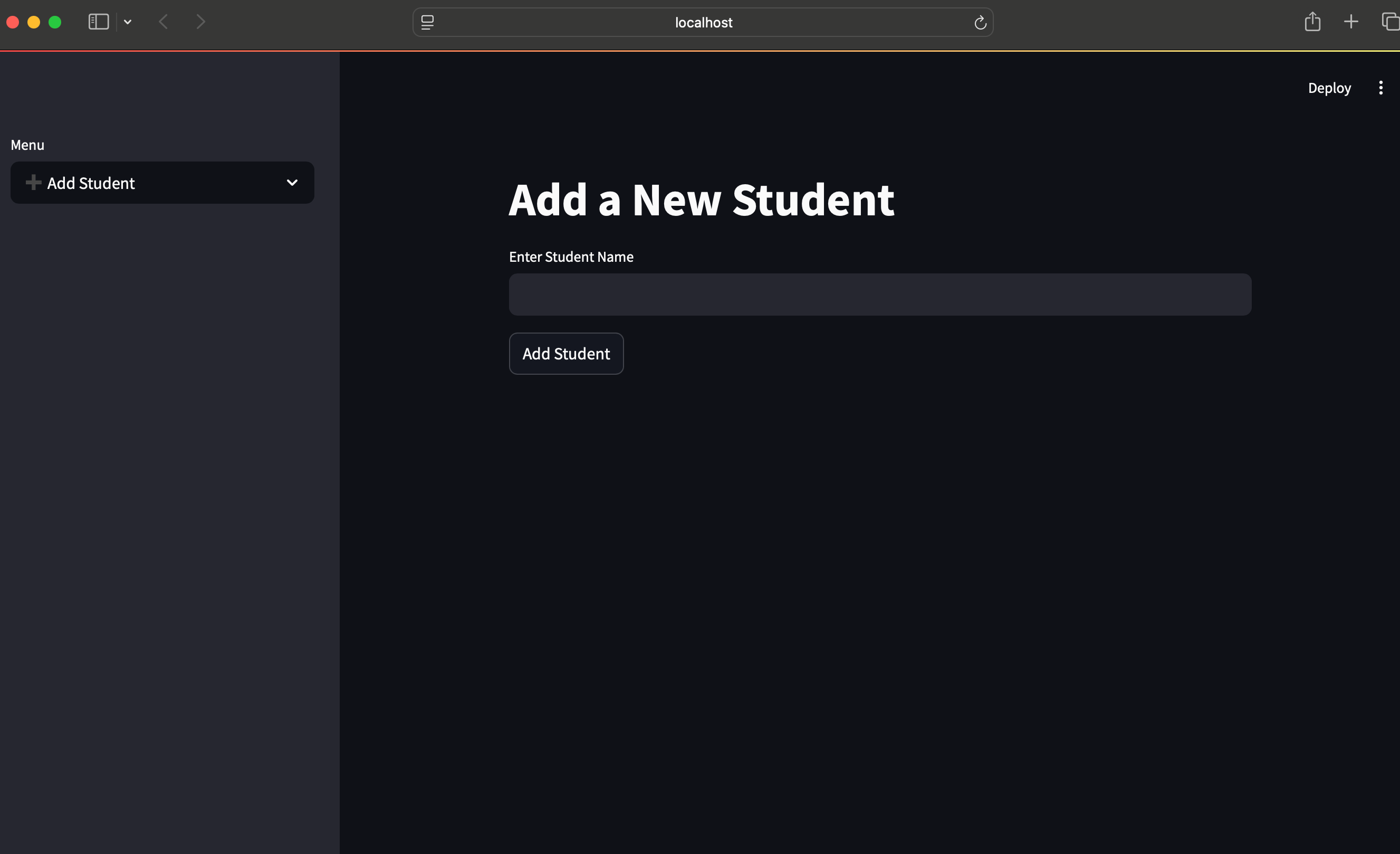
st.dataframe(filtered\_summary)

else:

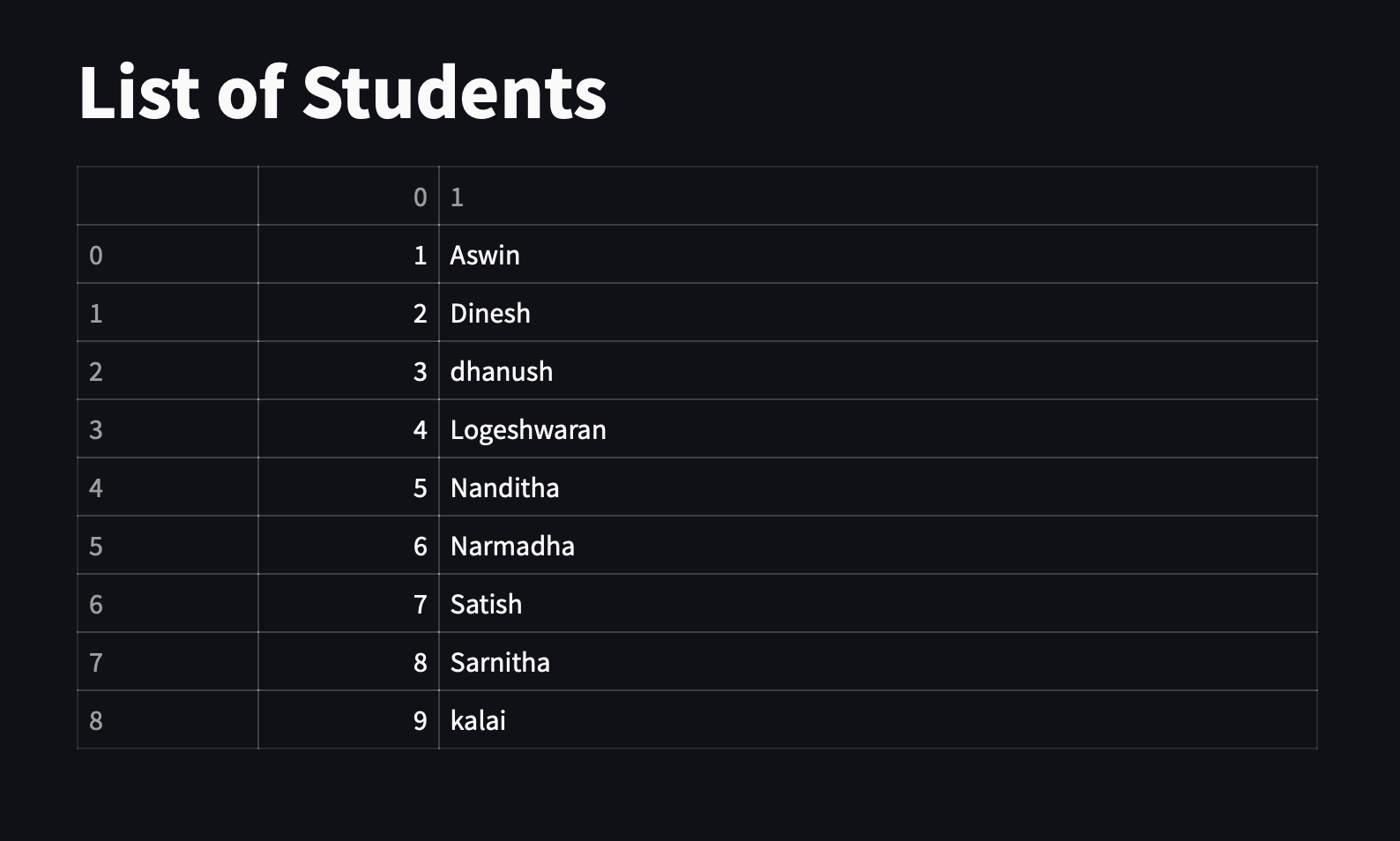
st.success("No students found below the selected threshold!")

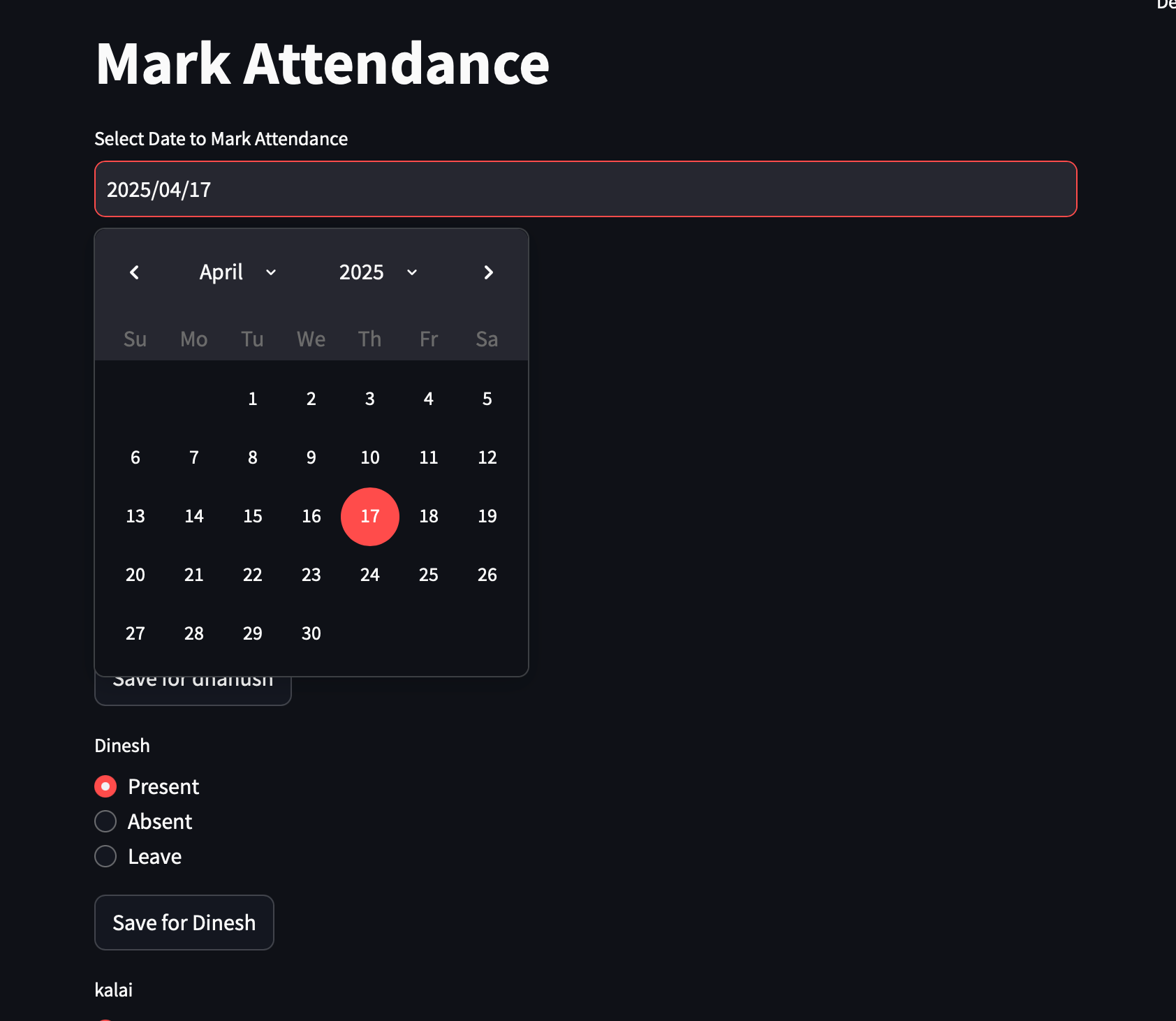
### 

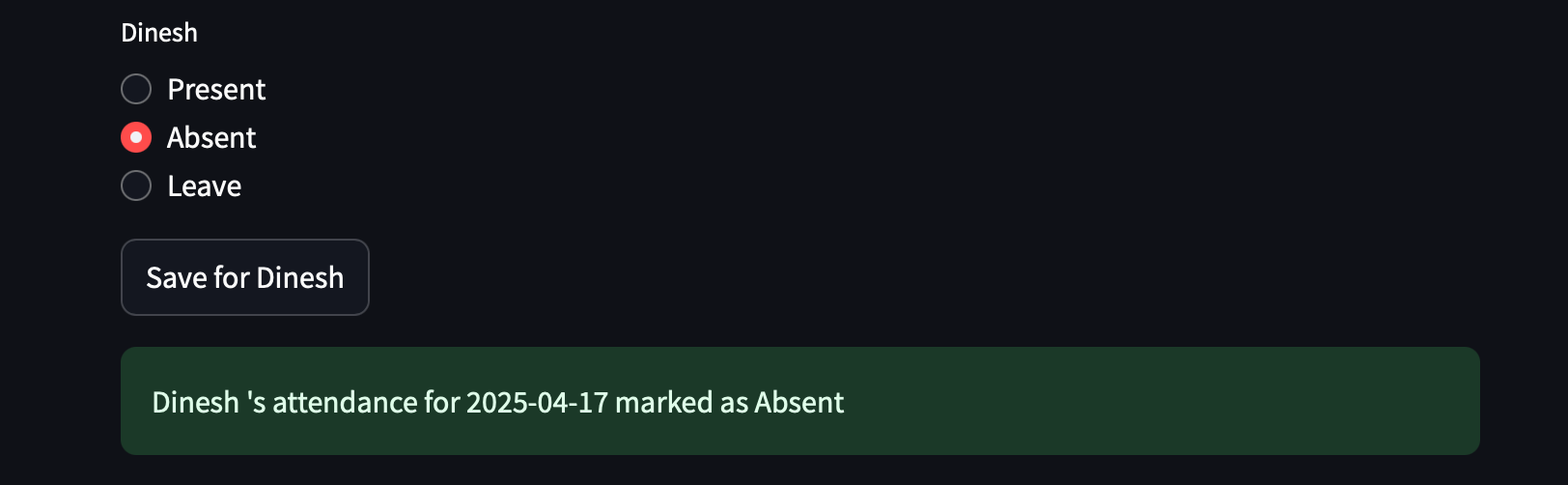
**Output:**

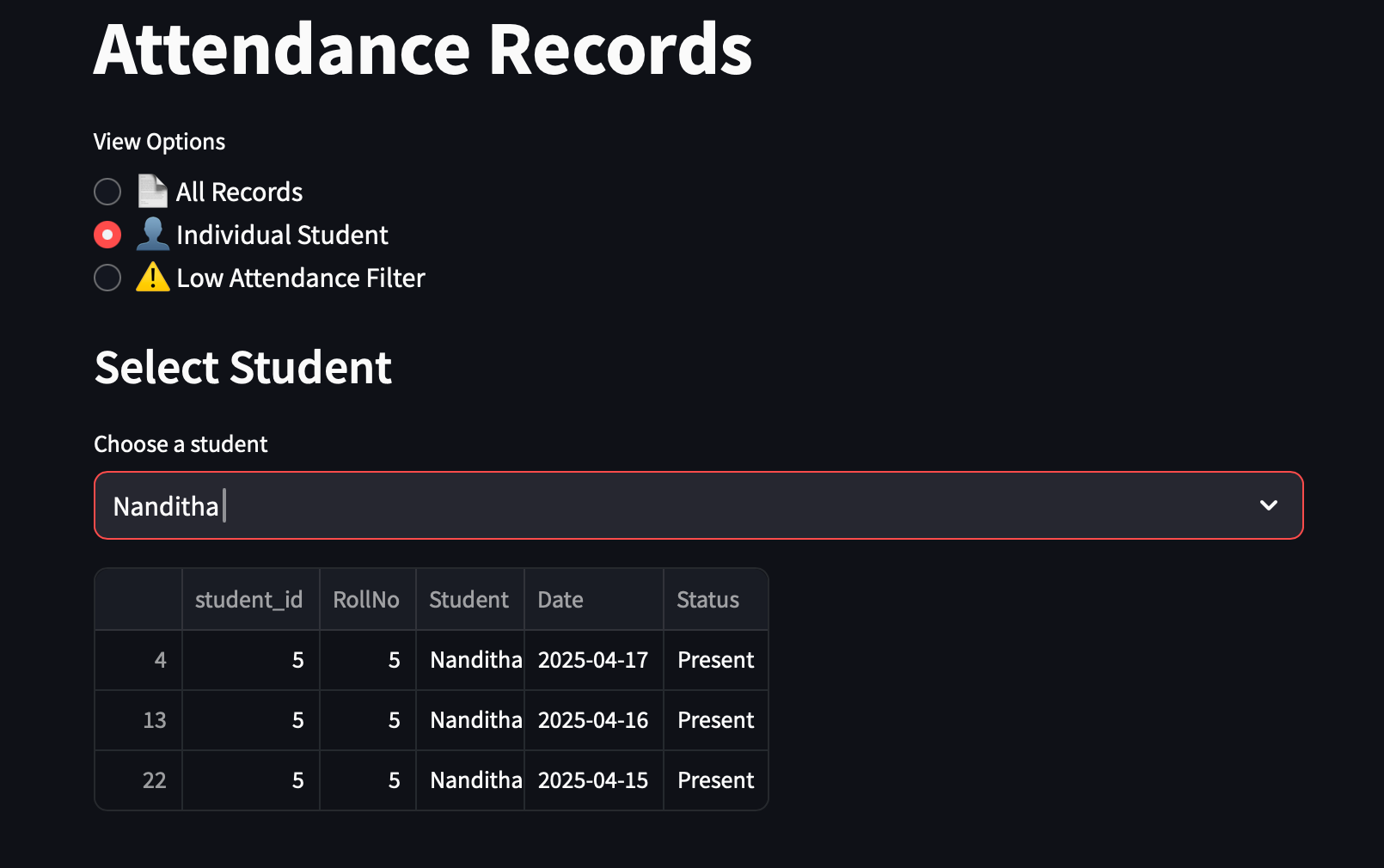


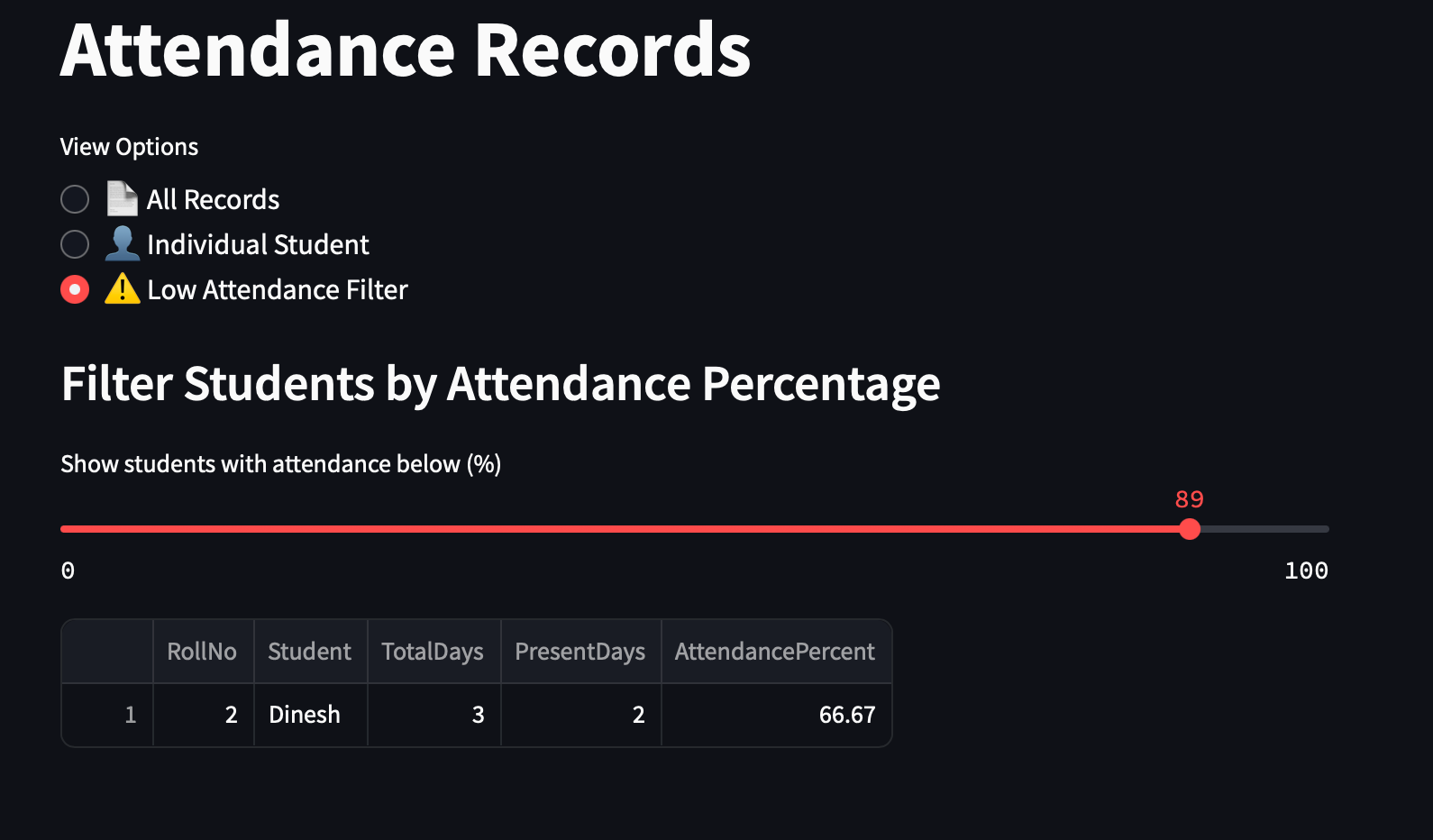
### 

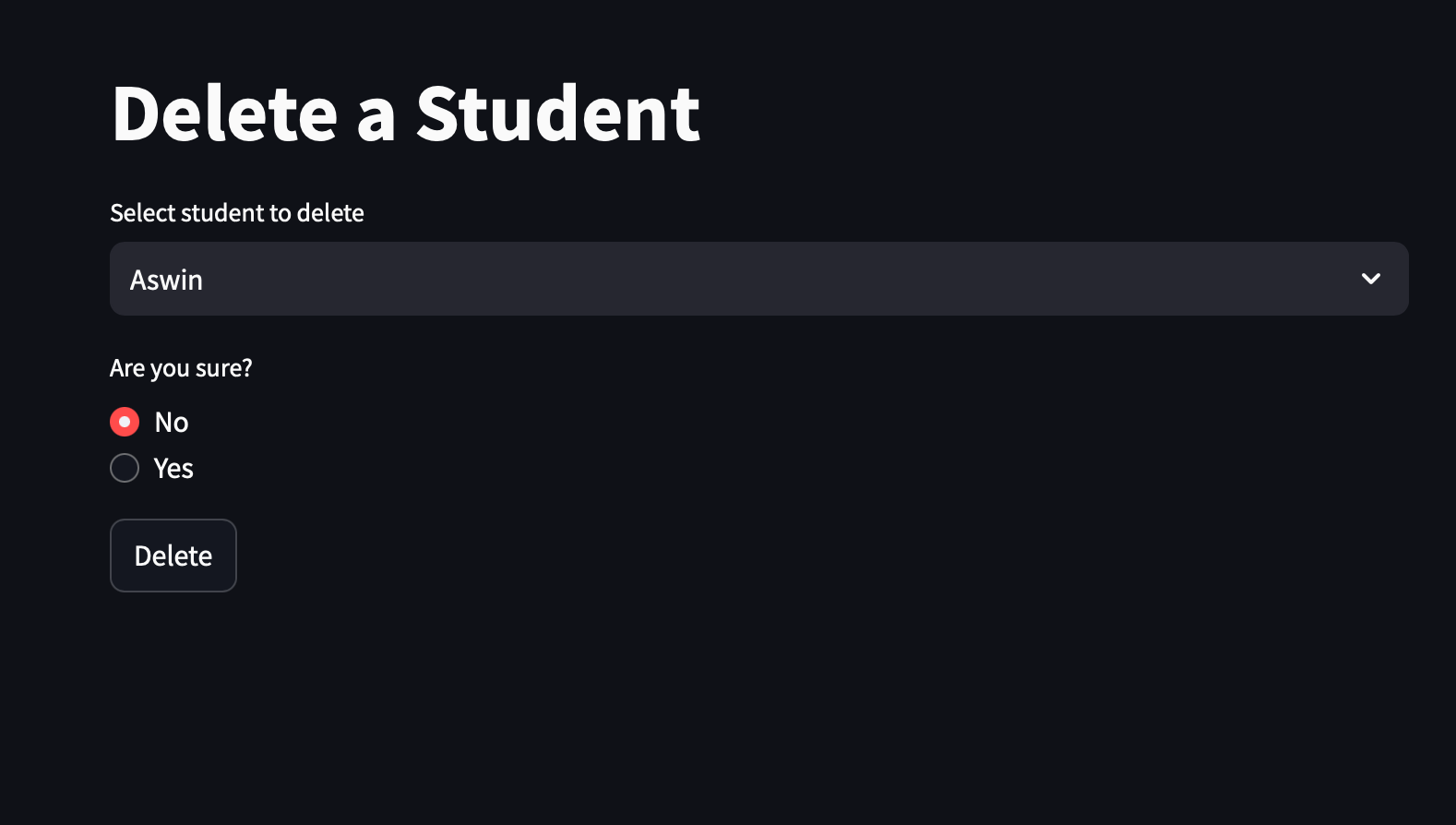


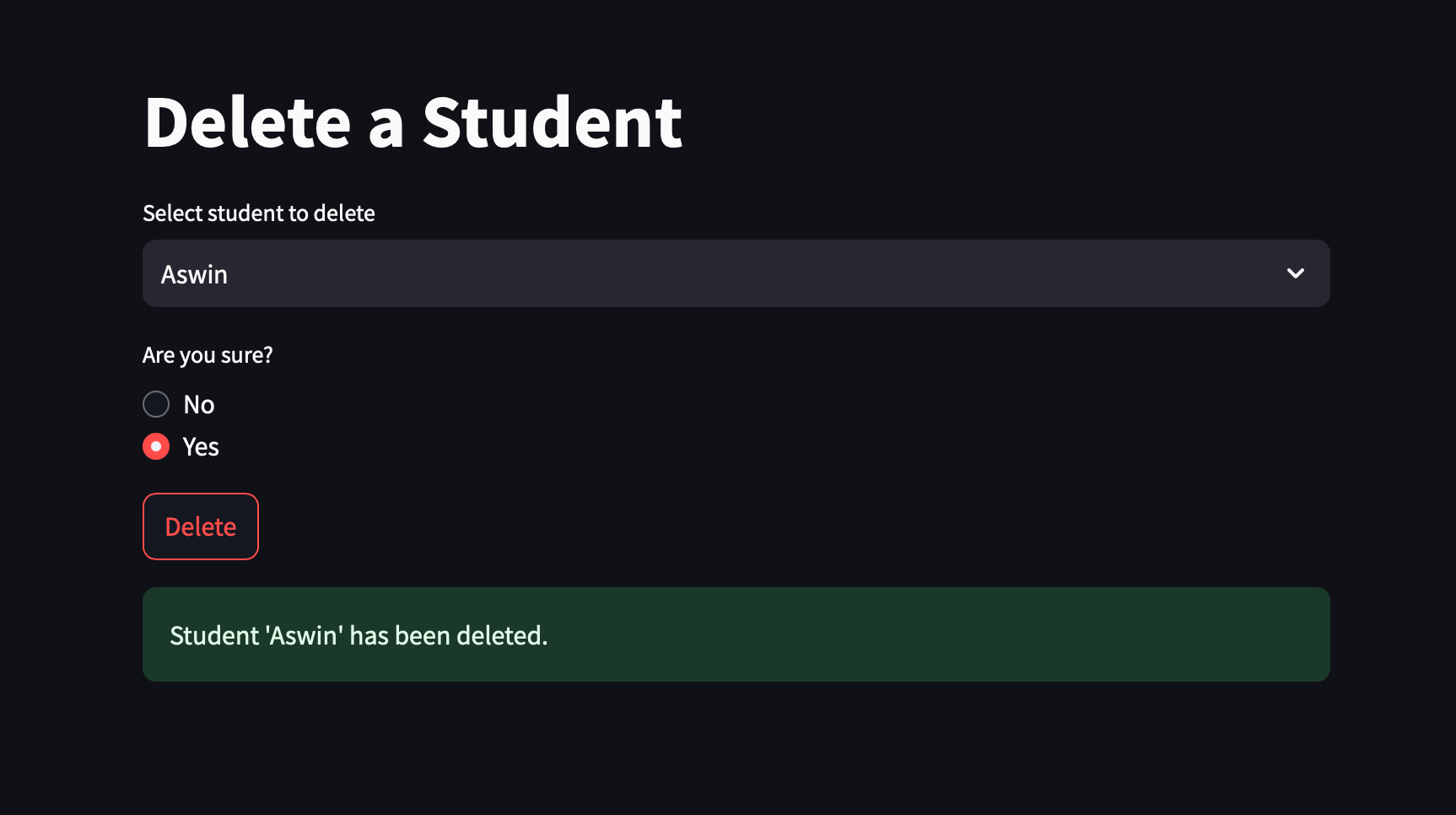












### **Result:**

Easy student record management

Daily attendance tracking

Identifies low attendance

Shows summaries for analysis

Saves all data securely in MySQL