import streamlit as st

import mysql.connector

import pandas as pd

# Function to create a connection to the database

def create\_connection():

    return mysql.connector.connect(

        host="localhost",

        user="root",

        password="12345678",

        database="student\_management"

    )

# Functions for user management

def create\_users\_table():

    connection = create\_connection()

    cursor = connection.cursor()

    cursor.execute("""

        CREATE TABLE IF NOT EXISTS users (

            username VARCHAR(50) PRIMARY KEY,

            password VARCHAR(50)

        )

    """)

    connection.commit()

    cursor.close()

    connection.close()

def add\_user(username, password):

    connection = create\_connection()

    cursor = connection.cursor()

    query = "INSERT INTO users (username, password) VALUES (%s, %s)"

    cursor.execute(query, (username, password))

    connection.commit()

    cursor.close()

    connection.close()

    st.success("User registered successfully.")

def validate\_user(username, password):

    connection = create\_connection()

    cursor = connection.cursor()

    query = "SELECT \* FROM users WHERE username = %s AND password = %s"

    cursor.execute(query, (username, password))

    user = cursor.fetchone()

    cursor.close()

    connection.close()

    if user:

        return True

    return False

# Functions for student management

def insert\_student(first\_name, last\_name, dob, gender, email, phone):

    connection = create\_connection()

    cursor = connection.cursor()

    query = "INSERT INTO students (first\_name, last\_name, date\_of\_birth, gender, email, phone\_number) VALUES (%s, %s, %s, %s, %s, %s)"

    cursor.execute(query, (first\_name, last\_name, dob, gender, email, phone))

    connection.commit()

    cursor.close()

    connection.close()

    st.success("Student inserted successfully.")

def update\_student(student\_id, first\_name, last\_name, dob, gender, email, phone):

    connection = create\_connection()

    cursor = connection.cursor()

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

    if cursor.fetchone():

        query = "UPDATE students SET first\_name=%s, last\_name=%s, date\_of\_birth=%s, gender=%s, email=%s, phone\_number=%s WHERE student\_id=%s"

        cursor.execute(query, (first\_name, last\_name, dob, gender, email, phone, student\_id))

        connection.commit()

        st.success("Student updated successfully.")

    else:

        st.error("Student ID not found.")

    cursor.close()

    connection.close()

def delete\_student(student\_id):

    connection = create\_connection()

    cursor = connection.cursor()

    # Check if the student exists

    cursor.execute("SELECT \* FROM combined\_data WHERE student\_id=%s", (student\_id,))

    if cursor.fetchone():

        # Delete from combined\_data table

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

        # Delete from attendance table

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

        # Delete from grades table

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

        # Delete from enrollments table

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

        # Finally, delete from students table

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

        connection.commit()

        st.success("Student and related data deleted successfully.")

    else:

        st.error("Student ID not found.")

    cursor.close()

    connection.close()

def show\_data(query):

    connection = create\_connection()

    cursor = connection.cursor()

    cursor.execute(query)

    rows = cursor.fetchall()

    columns = [desc[0] for desc in cursor.description]

    cursor.close()

    connection.close()

    return rows, columns

def insert\_course(course\_id, course\_name, description, credits):

    connection = create\_connection()

    cursor = connection.cursor()

    query = "INSERT INTO courses (course\_id, course\_name, description, credits) VALUES (%s, %s, %s, %s)"

    cursor.execute(query, (course\_id, course\_name, description, credits))

    connection.commit()

    cursor.close()

    connection.close()

    st.success("Course inserted successfully.")

def enroll\_student(enrollment\_id, student\_id, course\_id, enrollment\_date):

    connection = create\_connection()

    cursor = connection.cursor()

    query = "INSERT INTO enrollments (enrollment\_id, student\_id, course\_id, enrollment\_date) VALUES (%s, %s, %s, %s)"

    cursor.execute(query, (enrollment\_id, student\_id, course\_id, enrollment\_date))

    connection.commit()

    cursor.close()

    connection.close()

    st.success("Enrolled successfully.")

def grade\_student(grade\_id, student\_id, course\_id, grade):

    connection = create\_connection()

    cursor = connection.cursor()

    query = "INSERT INTO grades (grade\_id, student\_id, course\_id, grade) VALUES (%s, %s, %s, %s)"

    cursor.execute(query, (grade\_id, student\_id, course\_id, grade))

    connection.commit()

    cursor.close()

    connection.close()

    st.success("Grade added successfully.")

def attend\_student(attendance\_id, student\_id, course\_id, attendance\_date, status):

    connection = create\_connection()

    cursor = connection.cursor()

    query = "INSERT INTO attendance (attendance\_id, student\_id, course\_id, attendance\_date, status) VALUES (%s, %s, %s, %s, %s)"

    cursor.execute(query, (attendance\_id, student\_id, course\_id, attendance\_date, status))

    connection.commit()

    cursor.close()

    connection.close()

    st.success("Attendance added successfully.")

def create\_combined\_table():

    connection = create\_connection()

    cursor = connection.cursor()

    cursor.execute("""

        CREATE TABLE IF NOT EXISTS combined\_data (

            student\_id INT,

            first\_name VARCHAR(50),

            last\_name VARCHAR(50),

            date\_of\_birth DATE,

            gender VARCHAR(10),

            email VARCHAR(100),

            phone\_number VARCHAR(20),

            course\_id VARCHAR(50),

            course\_name VARCHAR(100),

            description TEXT,

            credits INT,

            enrollment\_id VARCHAR(50),

            enrollment\_date DATE,

            grade\_id VARCHAR(50),

            grade VARCHAR(10),

            attendance\_id VARCHAR(50),

            attendance\_date DATE,

            status VARCHAR(20)

        )

    """)

    connection.commit()

    cursor.close()

    connection.close()

def insert\_combined\_data():

    connection = create\_connection()

    cursor = connection.cursor()

    cursor.execute("TRUNCATE TABLE combined\_data")

    cursor.execute("""

        INSERT INTO combined\_data (student\_id, first\_name, last\_name, date\_of\_birth, gender, email, phone\_number)

        SELECT student\_id, first\_name, last\_name, date\_of\_birth, gender, email, phone\_number FROM students

    """)

    cursor.execute("""

        UPDATE combined\_data

        INNER JOIN enrollments ON combined\_data.student\_id = enrollments.student\_id

        SET combined\_data.course\_id = enrollments.course\_id, combined\_data.enrollment\_id = enrollments.enrollment\_id, combined\_data.enrollment\_date = enrollments.enrollment\_date

    """)

    cursor.execute("""

        UPDATE combined\_data

        INNER JOIN courses ON combined\_data.course\_id = courses.course\_id

        SET combined\_data.course\_name = courses.course\_name, combined\_data.description = courses.description, combined\_data.credits = courses.credits

    """)

    cursor.execute("""

        UPDATE combined\_data

        INNER JOIN grades ON combined\_data.student\_id = grades.student\_id AND combined\_data.course\_id = grades.course\_id

        SET combined\_data.grade\_id = grades.grade\_id, combined\_data.grade = grades.grade

    """)

    cursor.execute("""

        UPDATE combined\_data

        INNER JOIN attendance ON combined\_data.student\_id = attendance.student\_id AND combined\_data.course\_id = attendance.course\_id

        SET combined\_data.attendance\_id = attendance.attendance\_id, combined\_data.attendance\_date = attendance.attendance\_date, combined\_data.status = attendance.status

    """)

    connection.commit()

    cursor.close()

    connection.close()

# Custom CSS for Google's color scheme

def add\_custom\_css():

    def add\_custom\_css():

        st.markdown(

            """

            <style>

            body {

                background-color: #2196F3; /\* Blue background \*/

            }

            .sidebar .sidebar-content {

                background-color: #2196F3str; /\* Black sidebar \*/

            }

            .css-1d391kg {background-color: white;}

            .css-18e3th9 {background-color: white;}

            .css-hxt7ib {background-color: white;}

            .css-1lcbmhc {background-color: white;}

            .css-2trqyj {background-color: white;}

            .css-2b097c-container {background-color: white;}

            .css-1fv8s86 {background-color: #4285F4; color: white;}

            .stButton>button {background-color: #4285F4; color: white;}

            .st-bf, .st-bk, .st-bc, .st-bs, .st-bx {background-color: white;}

            .css-1v0mbdj {background-color: white;}

            </style>

            """,

            unsafe\_allow\_html=True

        )

# Main application

def main():

    add\_custom\_css()

    create\_users\_table()

    st.title("Student Management System")

    if "authenticated" not in st.session\_state:

        st.session\_state.authenticated = False

    if not st.session\_state.authenticated:

        st.title("Login")

        choice = st.selectbox("Login/Sign Up", ["Login", "Sign Up"])

        if choice == "Login":

            username = st.text\_input("Username")

            password = st.text\_input("Password", type="password")

            if st.button("Login"):

                if validate\_user(username, password):

                    st.session\_state.authenticated = True

                    st.success("Login successful")

                else:

                    st.error("Invalid username or password")

        elif choice == "Sign Up":

            username = st.text\_input("New Username")

            password = st.text\_input("New Password", type="password")

            if st.button("Sign Up"):

                add\_user(username, password)

        return

    menu = ["Add Data", "Show Data", "Update Data", "Delete Data", "All Data"]

    choice = st.sidebar.selectbox("Menu", menu)

    if choice == "Add Data":

        st.subheader("Add Data")

        add\_choice = st.selectbox("Choose Data to Add", ["Student", "Course", "Enrollment", "Grade", "Attendance"])

        if add\_choice == "Student":

            first\_name = st.text\_input("First Name")

            last\_name = st.text\_input("Last Name")

            dob = st.date\_input("Date of Birth")

            gender = st.selectbox("Gender", ["Male", "Female", "Other"])

            email = st.text\_input("Email")

            phone = st.text\_input("Phone Number")

            if st.button("Insert Student"):

                insert\_student(first\_name, last\_name, dob, gender, email, phone)

        elif add\_choice == "Course":

            course\_id = st.text\_input("Course ID")

            course\_name = st.text\_input("Course Name")

            description = st.text\_input("Description")

            credits = st.number\_input("Credits", min\_value=1, max\_value=5)

            if st.button("Insert Course"):

                insert\_course(course\_id, course\_name, description, credits)

        elif add\_choice == "Enrollment":

            enrollment\_id = st.text\_input("Enrollment ID")

            student\_id = st.text\_input("Student ID")

            course\_id = st.text\_input("Course ID")

            enrollment\_date = st.date\_input("Enrollment Date")

            if st.button("Enroll Student"):

                enroll\_student(enrollment\_id, student\_id, course\_id, enrollment\_date)

        elif add\_choice == "Grade":

            grade\_id = st.text\_input("Grade ID")

            student\_id = st.text\_input("Student ID")

            course\_id = st.text\_input("Course ID")

            grade = st.text\_input("Grade")

            if st.button("Add Grade"):

                grade\_student(grade\_id, student\_id, course\_id, grade)

        elif add\_choice == "Attendance":

            attendance\_id = st.text\_input("Attendance ID")

            student\_id = st.text\_input("Student ID")

            course\_id = st.text\_input("Course ID")

            attendance\_date = st.date\_input("Attendance Date")

            status = st.selectbox("Status", ["Present", "Absent"])

            if st.button("Add Attendance"):

                attend\_student(attendance\_id, student\_id, course\_id, attendance\_date, status)

    elif choice == "Show Data":

        st.subheader("Show Data")

        show\_choice = st.selectbox("Choose Data to Show",

                                   ["Students", "Courses", "Enrollments", "Grades", "Attendance"])

        search\_value = st.text\_input("Search")

        if show\_choice == "Students":

            query = "SELECT \* FROM students"

            if search\_value:

                query += f" WHERE CONCAT\_WS(' ', student\_id, first\_name, last\_name, date\_of\_birth, gender, email, phone\_number) LIKE '%{search\_value}%'"

            data, columns = show\_data(query)

        elif show\_choice == "Courses":

            query = "SELECT \* FROM courses"

            if search\_value:

                query += f" WHERE CONCAT\_WS(' ', course\_id, course\_name, description, credits) LIKE '%{search\_value}%'"

            data, columns = show\_data(query)

        elif show\_choice == "Enrollments":

            query = "SELECT \* FROM enrollments"

            if search\_value:

                query += f" WHERE CONCAT\_WS(' ', enrollment\_id, student\_id, course\_id, enrollment\_date) LIKE '%{search\_value}%'"

            data, columns = show\_data(query)

        elif show\_choice == "Grades":

            query = "SELECT \* FROM grades"

            if search\_value:

                query += f" WHERE CONCAT\_WS(' ', grade\_id, student\_id, course\_id, grade) LIKE '%{search\_value}%'"

            data, columns = show\_data(query)

        elif show\_choice == "Attendance":

            query = "SELECT \* FROM attendance"

            if search\_value:

                query += f" WHERE CONCAT\_WS(' ', attendance\_id, student\_id, course\_id, attendance\_date, status) LIKE '%{search\_value}%'"

            data, columns = show\_data(query)

        if data:

            st.dataframe(pd.DataFrame(data, columns=columns))

        else:

            st.write("No data found")

    elif choice == "Update Data":

        st.subheader("Update Data")

        student\_data, columns = show\_data("SELECT student\_id, first\_name, last\_name FROM students")

        if student\_data:

            student\_dict = {f"{student[1]} {student[2]} ({student[0]})": student[0] for student in student\_data}

            student\_id = st.selectbox("Select Student", list(student\_dict.keys()))

            student\_id = student\_dict[student\_id]

            first\_name = st.text\_input("First Name")

            last\_name = st.text\_input("Last Name")

            dob = st.date\_input("Date of Birth")

            gender = st.selectbox("Gender", ["Male", "Female", "Other"])

            email = st.text\_input("Email")

            phone = st.text\_input("Phone Number")

            if st.button("Update Student"):

                update\_student(student\_id, first\_name, last\_name, dob, gender, email, phone)

        else:

            st.write("No students found")

    elif choice == "Delete Data":

        st.subheader("Delete Data")

        student\_data, columns = show\_data("SELECT student\_id, first\_name, last\_name FROM students")

        if student\_data:

            student\_dict = {f"{student[1]} {student[2]} ({student[0]})": student[0] for student in student\_data}

            student\_id = st.selectbox("Select Student", list(student\_dict.keys()))

            student\_id = student\_dict[student\_id]

            if st.button("Delete Student"):

                delete\_student(student\_id)

        else:

            st.write("No students found")

    elif choice == "All Data":

        st.subheader("All Data")

        create\_combined\_table()

        insert\_combined\_data()

        search\_value = st.text\_input("Search by student\_id, name, or date of birth")

        query = "SELECT \* FROM combined\_data"

        if search\_value:

            query += f" WHERE CONCAT\_WS(' ', student\_id, first\_name, last\_name, date\_of\_birth) LIKE '%{search\_value}%'"

        data, columns = show\_data(query)

        if data:

            st.dataframe(pd.DataFrame(data, columns=columns))

        else:

            st.write("No data found")

    if st.sidebar.button("Logout"):

        st.session\_state.authenticated = False

if \_\_name\_\_ == "\_\_main\_\_":

    main()