```
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-1vOmbdj {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()
```