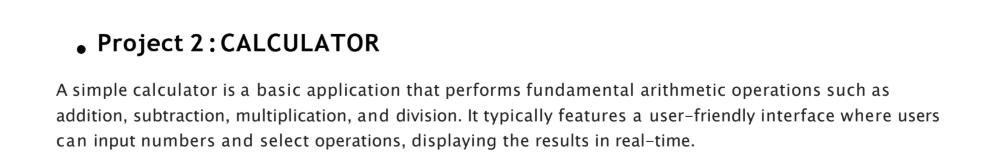
**NAME : a . mayank**

**E MAIL :** [**mayank.atmakuri@aurora.edu.in**](mailto:mayank.atmakuri@aurora.edu.in)

**Phone no : 9014227196**





This is the main app.py code

from flask import Flask, render\_template, request

# Initialize the Flask application

app = Flask(\_\_name\_\_)

@app.route('/', methods=['GET', 'POST'])

def calculator():

    """

    Main route for the calculator.

    Handles both the initial page load (GET) and form submissions (POST).

    """

    result = "" # Initialize result to an empty string

    if request.method == 'POST':

        # request.form is a dictionary-like object containing the form data

        expression = request.form.get('expression', '')

        try:

            # IMPORTANT: Using eval() is a security risk in a real public application

            # as it can execute arbitrary code. For this simple, local project, it's fine.

            if expression:

                result = str(eval(expression))

            else:

                result = "Error: No expression provided"

        except ZeroDivisionError:

            result = "Error: Cannot divide by zero"

        except Exception as e:

            # Catch other potential errors like invalid syntax

            result = "Error: Invalid Expression"

    # Render the HTML template and pass the result to it

    return render\_template('index.html', result=result)

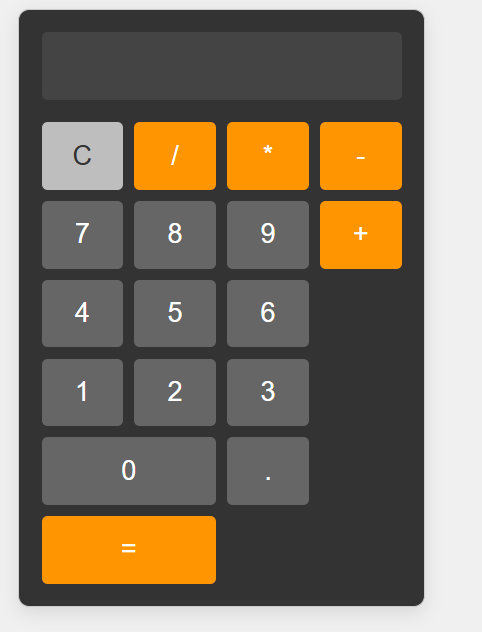
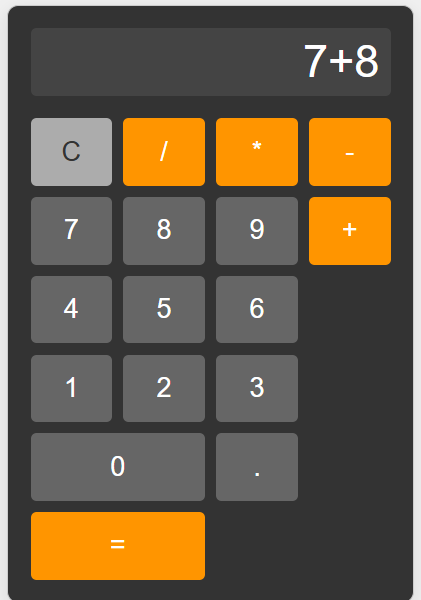
# This allows you to run the app directly from the Python script

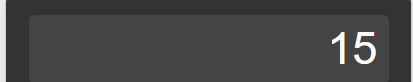
if \_\_name\_\_ == '\_\_main\_\_':

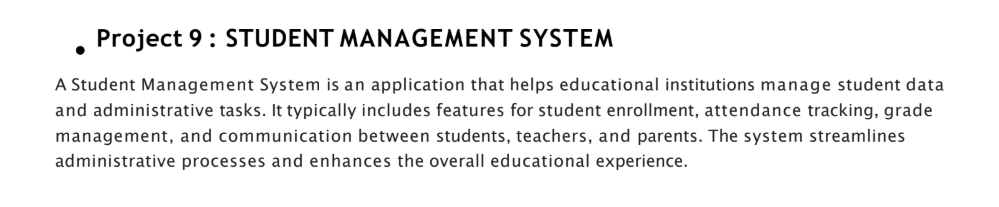
    # debug=True will auto-reload the server when you save changes

    app.run(debug=True)

**out put**

****

****

****

app.py code

from flask import Flask, render\_template, request, redirect, url\_for, flash, session, send\_file

import sqlite3

from datetime import datetime

from io import BytesIO

import openpyxl

app = Flask(\_\_name\_\_)

app.secret\_key = 'my-secret-key'

# ---------------- INIT DB ----------------

def init\_db():

    conn = sqlite3.connect('database.db')

    cursor = conn.cursor()

    cursor.execute('''

        CREATE TABLE IF NOT EXISTS students (

            id INTEGER PRIMARY KEY AUTOINCREMENT,

            name TEXT NOT NULL,

            email TEXT,

            phone TEXT,

            grade TEXT

        )

    ''')

    # ### CHANGE 1: Added a UNIQUE constraint.

    # ### This is crucial. It ensures a student can only have one attendance status per day.

    # ### It allows us to use "INSERT OR REPLACE" to easily update today's status.

    cursor.execute('''

        CREATE TABLE IF NOT EXISTS attendance (

            id INTEGER PRIMARY KEY AUTOINCREMENT,

            student\_id INTEGER,

            date TEXT,

            status TEXT,

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

            UNIQUE(student\_id, date)

        )

    ''')

    conn.commit()

    conn.close()

# ---------------- LOGIN ----------------

@app.route('/login', methods=['GET', 'POST'])

def login():

    if request.method == 'POST':

        username = request.form['username']

        password = request.form['password']

        if username == 'admin' and password == 'admin123':

            session['admin'] = True

            flash('Login successful!', 'success')

            return redirect(url\_for('index'))

        else:

            flash('Invalid credentials!', 'danger')

            return redirect(url\_for('login'))

    return render\_template('login.html')

@app.route('/logout')

def logout():

    session.pop('admin', None)

    flash("Logged out successfully!", 'info')

    return redirect(url\_for('login'))

# ---------------- HOME ----------------

@app.route('/')

def index():

    if not session.get('admin'):

        return redirect(url\_for('login'))

    conn = sqlite3.connect('database.db')

    conn.row\_factory = sqlite3.Row # Allows accessing columns by name

    cur = conn.cursor()

    # ### CHANGE 2: Updated the main query to get today's attendance status.

    # ### We use a LEFT JOIN to fetch the status from the 'attendance' table for today's date.

    # ### If a student hasn't been marked today, their status will be NULL (None in Python).

    today\_date = datetime.now().strftime('%Y-%m-%d')

    cur.execute("""

        SELECT s.id, s.name, s.email, s.phone, s.grade, a.status

        FROM students s

        LEFT JOIN attendance a ON s.id = a.student\_id AND a.date = ?

        ORDER BY s.id

    """, (today\_date,))

    students = cur.fetchall()

    conn.close()

    # ### Your HTML file needs to be named 'index.html' for this to work.

    # ### The HTML you provided earlier is perfect for this.

    return render\_template('index.html', students=students)

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

@app.route('/add', methods=['POST'])

def add\_student():

    if not session.get('admin'):

        return redirect(url\_for('login'))

    name = request.form['name']

    email = request.form['email']

    phone = request.form['phone']

    grade = request.form['grade']

    conn = sqlite3.connect('database.db')

    conn.execute("INSERT INTO students (name, email, phone, grade) VALUES (?, ?, ?, ?)",

                 (name, email, phone, grade))

    conn.commit()

    conn.close()

    flash("Student added!", 'success')

    return redirect(url\_for('index'))

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

@app.route('/delete/<int:id>')

def delete\_student(id):

    if not session.get('admin'):

        return redirect(url\_for('login'))

    conn = sqlite3.connect('database.db')

    # Also deletes related attendance records because of "ON DELETE CASCADE" in the table definition

    conn.execute("DELETE FROM students WHERE id = ?", (id,))

    conn.commit()

    conn.close()

    flash("Student deleted!", 'success')

    return redirect(url\_for('index'))

# ---------------- EDIT STUDENT ----------------

@app.route('/edit/<int:id>', methods=['GET', 'POST'])

def edit\_student(id):

    if not session.get('admin'):

        return redirect(url\_for('login'))

    conn = sqlite3.connect('database.db')

    cur = conn.cursor()

    if request.method == 'POST':

        name = request.form['name']

        email = request.form['email']

        phone = request.form['phone']

        grade = request.form['grade']

        cur.execute("UPDATE students SET name=?, email=?, phone=?, grade=? WHERE id=?",

                    (name, email, phone, grade, id))

        conn.commit()

        conn.close()

        # ### CHANGE 3: Changed flash message to be more specific.

        flash("Student updated successfully!", 'success')

        return redirect(url\_for('index'))

    else:

        cur.execute("SELECT \* FROM students WHERE id = ?", (id,))

        student = cur.fetchone()

        conn.close()

        return render\_template('edit.html', student=student)

# ---------------- ATTENDANCE SYSTEM ROUTES ----------------

# ### CHANGE 4: This is the NEW route that fixes your "Not Found" error.

# ### It matches the URL from the dashboard buttons (e.g., /mark\_attendance/5/Present).

@app.route('/mark\_attendance/<int:student\_id>/<string:status>')

def mark\_attendance\_from\_dashboard(student\_id, status):

    if not session.get('admin'):

        return redirect(url\_for('login'))

    # Ensure status is only 'Present' or 'Absent' to be safe

    if status not in ['Present', 'Absent']:

        flash('Invalid attendance status provided.', 'danger')

        return redirect(url\_for('index'))

    today\_date = datetime.now().strftime('%Y-%m-%d')

    try:

        conn = sqlite3.connect('database.db')

        cursor = conn.cursor()

        # Use "INSERT OR REPLACE" thanks to the UNIQUE constraint.

        # It will INSERT a new row if one doesn't exist for the student/date.

        # It will REPLACE the old row if one already exists.

        cursor.execute("""

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

            VALUES (?, ?, ?)

        """, (student\_id, today\_date, status))

        conn.commit()

        conn.close()

        flash(f'Attendance for student marked as {status}.', 'success')

    except Exception as e:

        flash(f'An error occurred: {e}', 'danger')

    return redirect(url\_for('index'))

# ### NOTE: Your other attendance routes are for a separate feature (bulk attendance page).

# ### I've left them here as they don't conflict.

@app.route('/attendance\_records')

def attendance\_records():

    if not session.get('admin'):

        return redirect(url\_for('login'))

    conn = sqlite3.connect('database.db')

    cur = conn.cursor()

    cur.execute('''

        SELECT s.name, a.date, a.status

        FROM attendance a

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

        ORDER BY a.date DESC, s.name ASC

    ''')

    records = cur.fetchall()

    conn.close()

    return render\_template('attendance\_records.html', records=records)

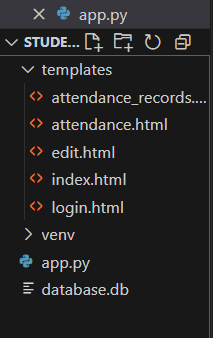
# ---------------- MAIN ----------------

if \_\_name\_\_ == '\_\_main\_\_':

    init\_db()

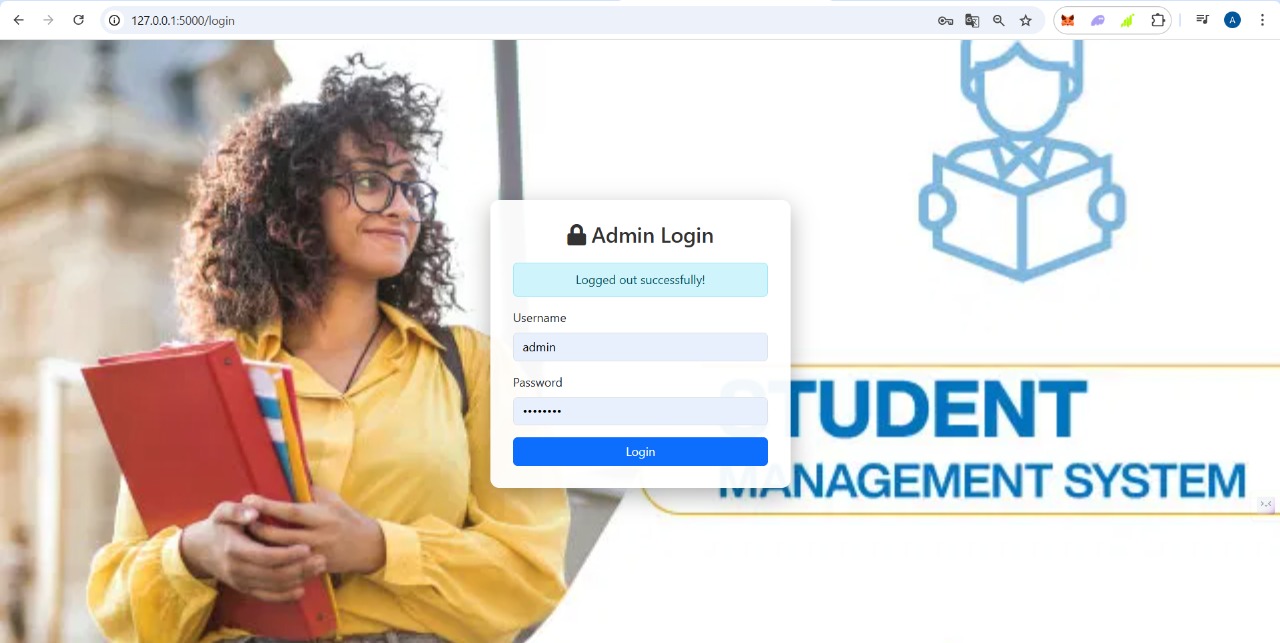
    app.run(debug=True)

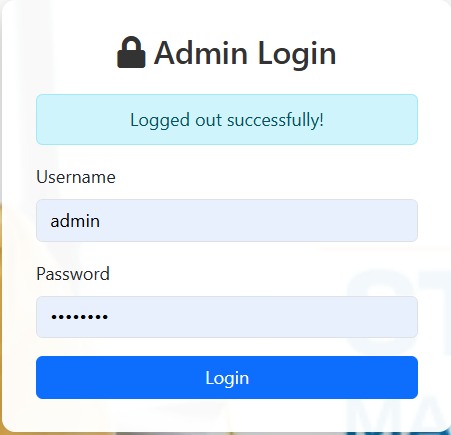
**templates floder in this all HTML codes are saved**



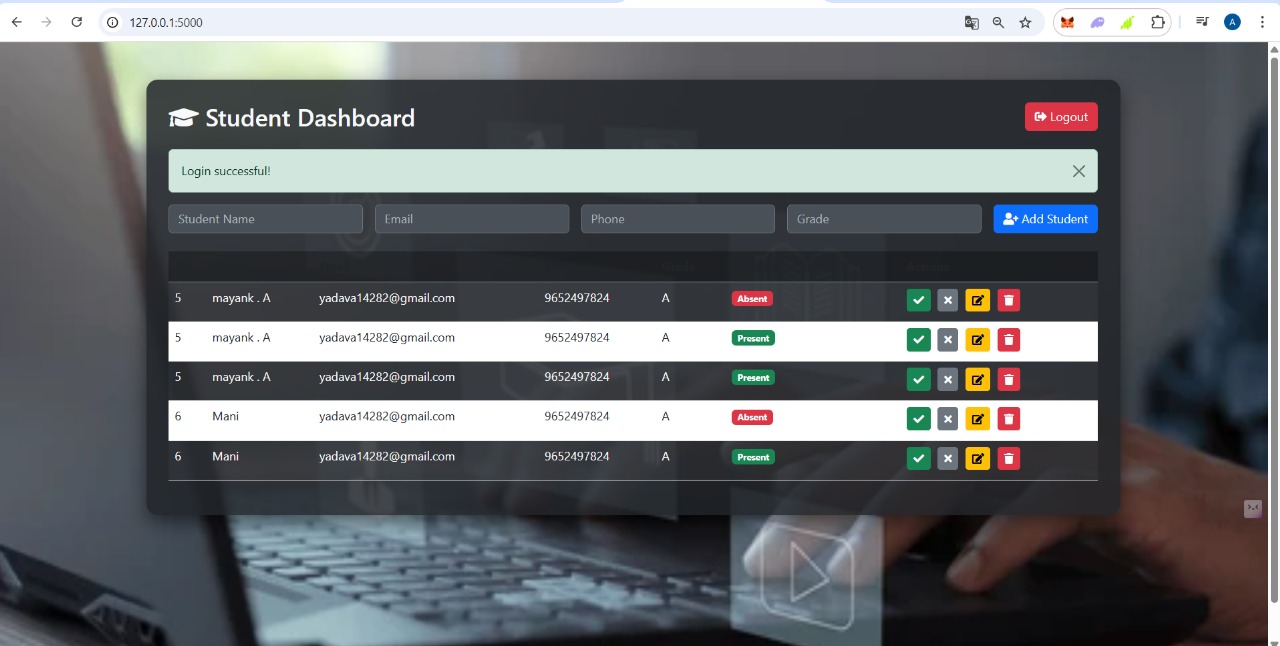
**Out put**

**Login page**

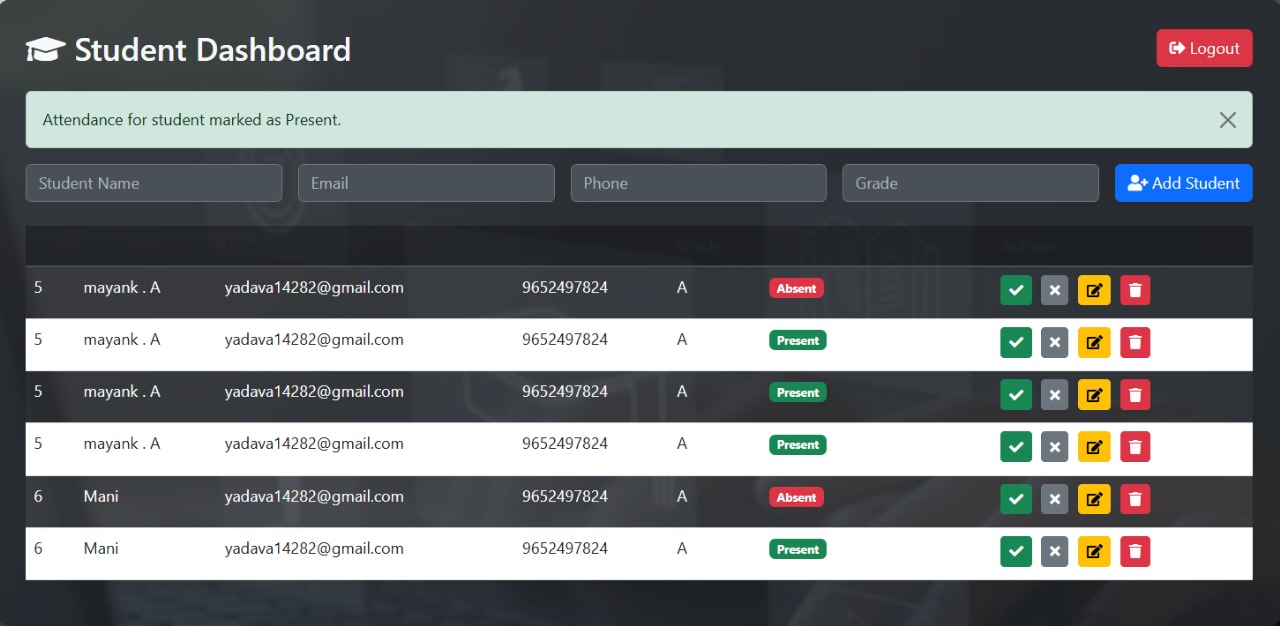




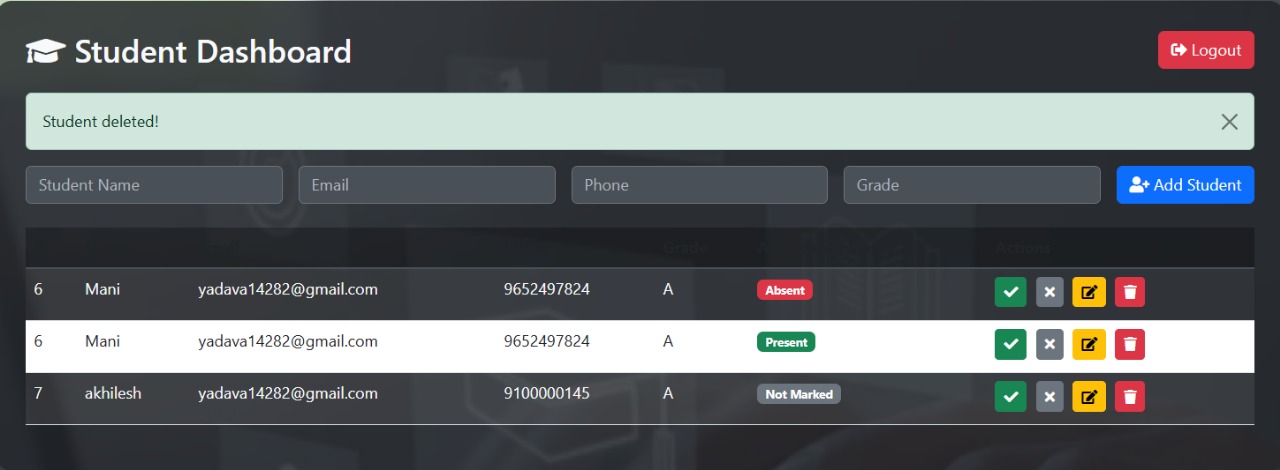
**Home page**



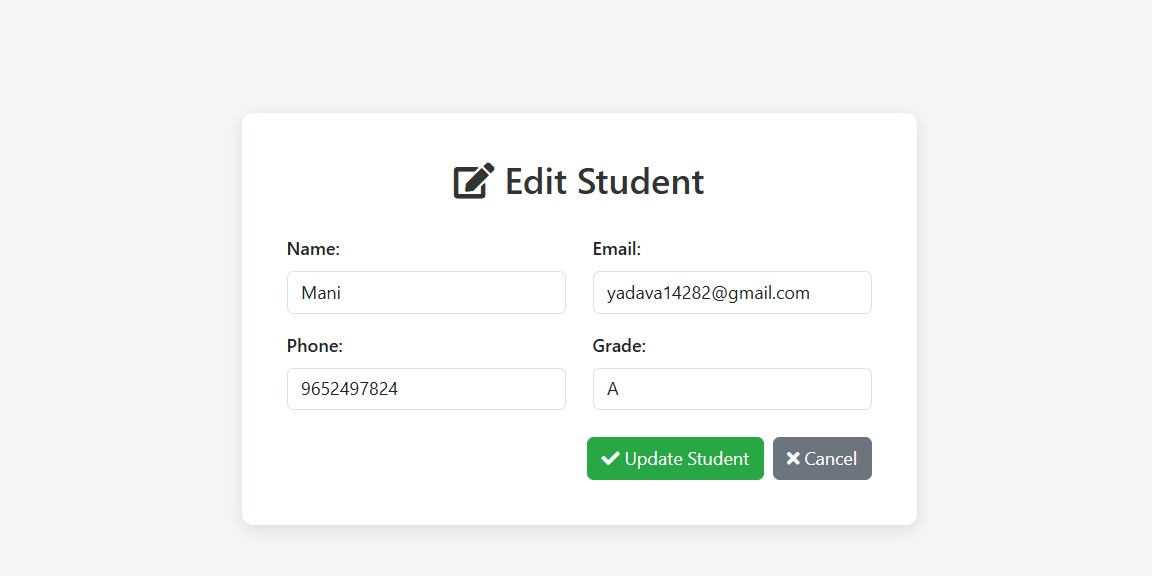
**Attendance marking :**



**Student deleted :**



**Edit student details :**



**🎓 Student Dashboard: Button Functionality Explained**

Each student row in your table includes **attendance status + four action buttons**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Icon/Button** | **Function** | **Route Triggered** | **Behavior** |
| ✅ Green Check (✔️) | **Mark Present** | /mark\_attendance/<id>/Present | Marks student as **Present** for today in DB. |
| ❌ Gray Cross (❌) | **Mark Absent** | /mark\_attendance/<id>/Absent | Marks student as **Absent** for today in DB. |
| ✏️ Yellow Pencil | **Edit Student** | /edit/<id> | Opens a form to **edit** student details (name, email, etc.). |
| 🗑️ Red Trash | **Delete Student** | /delete/<id> | Permanently deletes student record + attendance. |

**🔍 Top Row Controls**

|  |  |
| --- | --- |
| **UI Element** | **Purpose** |
| 🔴 **Logout Button** | Logs the admin out and redirects to login page. |
| 🔵 **Add Student Button** | Opens form (or popup) to add a new student. |
| 🔎 **Search Fields (Name, Email, etc.)** | Likely connected to JS for filtering/searching the table dynamically (client-side). |

**🟢 Attendance Status Badges**

|  |  |
| --- | --- |
| **Badge** | **Meaning** |
| 🟩 Present (Green Badge) | Student is marked present **today**. |
| 🟥 Absent (Red Badge) | Student is marked absent **today**. |
| 🔲 *(Blank)* | No attendance data for today. |

The logic comes from this SQL query in your / route:

SELECT s.id, s.name, s.email, s.phone, s.grade, a.status

FROM students s

LEFT JOIN attendance a ON s.id = a.student\_id AND a.date = ?

It uses LEFT JOIN to pull attendance **only for today's date**.

**Technical Flow**

**1. Mark Attendance**

* Uses INSERT OR REPLACE INTO attendance to update status for the student for the current date.
* Ensured by UNIQUE(student\_id, date) constraint in DB.

**2. Delete Student**

* Deletes student from students table.
* Due to ON DELETE CASCADE, their attendance records are also deleted.

**3. Edit Student**

* Loads current student data in form.
* On POST, updates values in the DB.

**Example**

For Student ID 6 (Mani):

* **1st Row**: Marked **Absent** (shows red badge).
* **2nd Row**: Marked **Present** (shows green badge).

This might be duplicate rows — ensure the SELECT is DISTINCT or clean duplicates in DB.