Case Study:
Smart Weather Station using Raspberry Pi

Objective:

Build a low-cost, real-time weather station using Raspberry Pi that collects environmental data (temperature, humidity, pressure), processes it locally, visualizes it on a web interface, and optionally pushes data to the cloud.

Components Required:

Component	Description
Raspberry Pi 4/5	Core computing device
DHT22 or BME280	Sensor for temperature and humidity
BMP180 (optional)	Sensor for atmospheric pressure
Jumper Wires	For GPIO connections
Breadboard	Prototyping circuit assembly
Internet Connection	For cloud connectivity
Optional: LCD or OLED Display	For local display

Key Concepts Taught:

Concept	Learning Outcome
SoC	Understand Pi as a System on Chip with CPU, RAM, GPIO
GPIO Communication	Use GPIO pins to interface with sensors
I2C/SPI Protocols	Learn how sensors communicate with Raspberry Pi
Sensor Integration	Read temperature, humidity, and pressure using Python

Concept	Learning Outcome
Edge Computing	Local processing of data (e.g., alert when temp > threshold)
Web Server with Flask	Serve real-time data via a local web page
Data Logging	Store data in CSV/SQLite for analysis
Cloud Connectivity (optional)	Send data to ThingSpeak or AWS IoT for remote monitoring
Visualization	Plot live graphs using Matplotlib or JavaScript (Plotly)
Power Management	Teach power supply needs and energy efficiency

---

Workflow:

1. **Sensor Setup:**
    - Connect DHT22 to GPIO pins of Raspberry Pi.
    - Use Python libraries like Adafruit_DHT or smbus2 for I2C sensors.
2. **Data Acquisition:**
    - Write Python script to read sensor values every 10 seconds.
    - Store in local variables and print to console.
3. **Local Processing (Edge Computing):**
    - Add a condition to check thresholds (e.g., Temp > 35°C).
    - Display alert on console or trigger LED buzzer.
4. **Web Interface with Flask:**
    - Display live temperature and humidity via HTML page.
    - Refresh using JavaScript every few seconds.
5. **Data Logging:**
    - Store timestamped sensor data in a .csv file or SQLite database.
6. **Cloud Integration (Optional):**
    - Use requests or MQTT to send data to:
        - **ThingSpeak**

        - **AWS IoT Core**
        - **Google Sheets via API**
7. **Graphical Visualization:**
    - Use Plotly or Matplotlib to visualize trends (e.g., temperature over time).

Project Output:

- A live web dashboard showing real-time temperature, humidity, and pressure.
- Alerts on threshold breach.
- Cloud-stored historical data.

Skills Students Learn:

- **Basic electronics and prototyping**
- **Python scripting and data handling**
- **Web development (HTML, Flask, JavaScript)**
- **IoT communication protocols**
- **Data visualization**
- **System integration and debugging**

Possible Extensions:

- Add camera to capture environmental visuals.
- Use machine learning model to predict weather trends.
- Add solar power for sustainability.
- Integrate voice assistant using Google Assistant API.