Smart Environment Monitoring System

Course 158335 – The Internet of Things and Cloud Computing (Spring 2025)

student name Yunfeng Li

student ID 22009169

1 Introduction

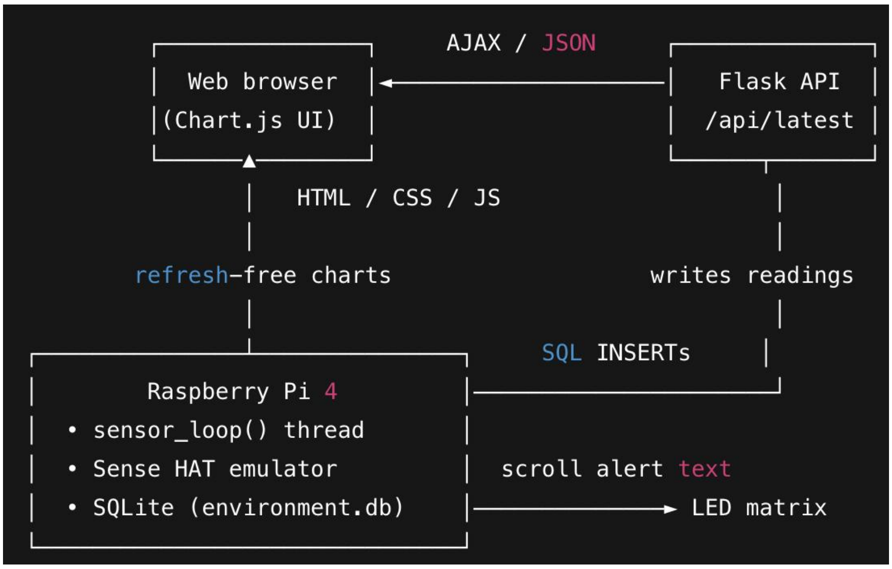
This Smart Environment Monitoring System turns a Raspberry Pi 4 (with a Sense HAT emulator) into a project that:

Every second, reads temperature, humidity, pressure.
stores every reading in an SQLite timeseries database;
It analyses the stream for spikes, trends and threshold breaches.
It visualises live and historical data on a Flask web dashboard.
It raises alarms on both the web UI and the Sense HAT LED matrix.

Project layout lee/

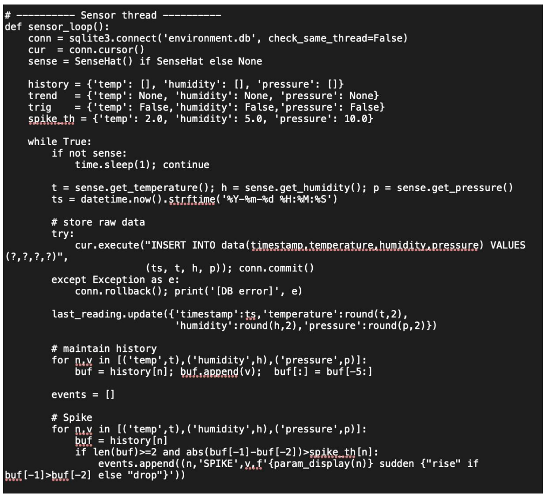
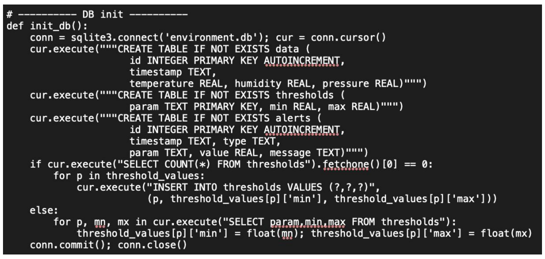
```
lee/
├─── app.py                  # Flask back-end main program
├─── templates/              # HTML template directory (Flask default)
│     ├─── realtime.html     # real-time data page template
│     ├─── history.html      # historical data page template
│     ├─── threshold.html    # threshold settings page template
│     └─── alerts.html       # alert log page template
├─── static/                 # static asset directory
│     └─── script.js         # front-end JavaScript for AJAX live updates
└─── environment.db          # SQLite database (auto-created on first run)
```

2 System Design & Architecture



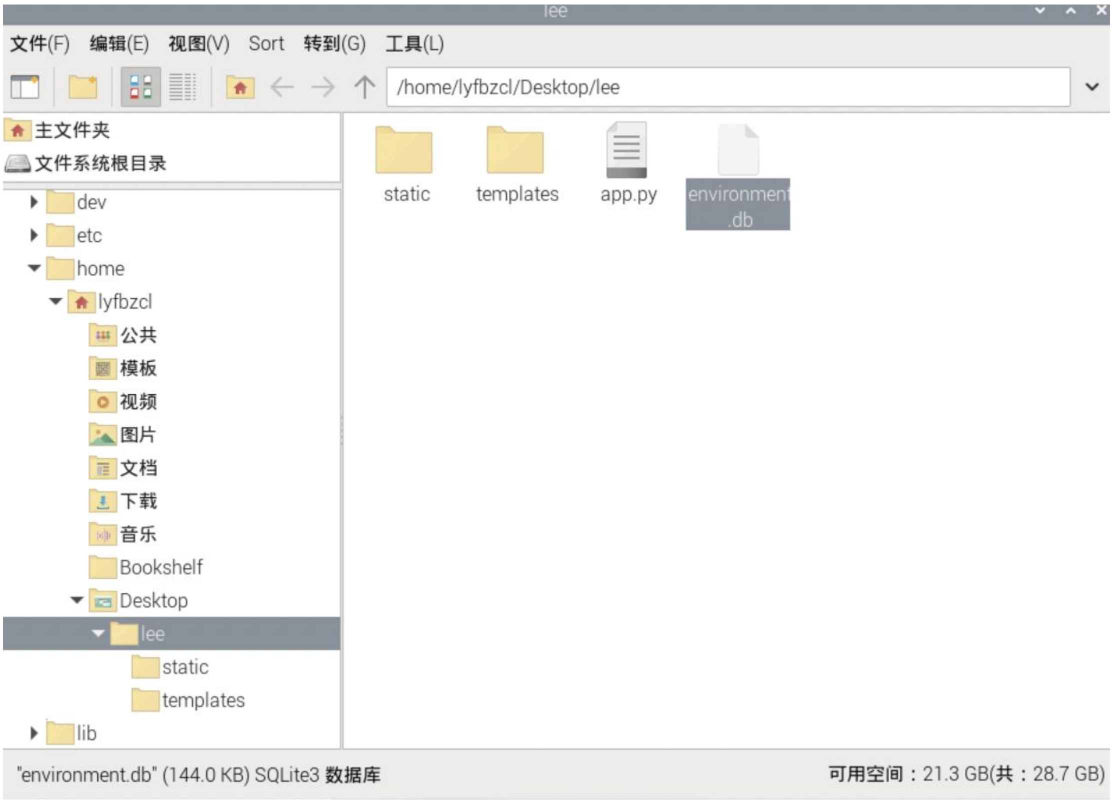
```
                              AJAX / JSON
           ┌──────────────┐                   ┌─────────────┐
           │ Web browser  │◄──────────────────│  Flask API  │
           │(Chart.js UI) │                   │ /api/latest │
           └──────▲───────┘                   └──────┬──────┘
                  │   HTML / CSS / JS                 │
                  │                                   │
        refresh-free charts                   writes readings
                  │                                   │
   ┌──────────────┴──────────────┐    SQL INSERTs     │
   │        Raspberry Pi 4       │────────────────────┘
   │  • sensor_loop() thread     │
   │  • Sense HAT emulator       │  scroll alert text
   │  • SQLite (environment.db)  │───────────► LED matrix
   └─────────────────────────────┘
```

3 Data Collection & Storage

Rubric line	Implementation
Accurate & reliable collection	sensor_loop() (lines 70-92) reads SenseHat.*() every second; values rounded to 0.01. 
Proper storage	Each reading inserted into **data** table with timestamp; init_db() creates tables if absent. 

```
# ---------- Sensor thread ----------
def sensor_loop():
    conn = sqlite3.connect('environment.db', check_same_thread=False)
    cur  = conn.cursor()
    sense = SenseHat() if SenseHat else None

    history = {'temp': [], 'humidity': [], 'pressure': []}
    trend   = {'temp': None, 'humidity': None, 'pressure': None}
    trig    = {'temp': False,'humidity': False,'pressure': False}
    spike_th = {'temp': 2.0, 'humidity': 5.0, 'pressure': 10.0}

    while True:
        if not sense:
            time.sleep(1); continue

        t = sense.get_temperature(); h = sense.get_humidity(); p = sense.get_pressure()
        ts = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        # store raw data
        try:
            cur.execute("INSERT INTO data(timestamp,temperature,humidity,pressure) VALUES
(?,?,?,?)",
                        (ts, t, h, p)); conn.commit()
        except Exception as e:
            conn.rollback(); print('[DB error]', e)

        last_reading.update({'timestamp':ts,'temperature':round(t,2),
                             'humidity':round(h,2),'pressure':round(p,2)})

        # maintain history
        for n,v in [('temp',t),('humidity',h),('pressure',p)]:
            buf = history[n]; buf.append(v);  buf[:] = buf[-5:]

        events = []

        # Spike
        for n,v in [('temp',t),('humidity',h),('pressure',p)]:
            buf = history[n]
            if len(buf)>=2 and abs(buf[-1]-buf[-2])>spike_th[n]:
                events.append((n,'SPIKE',v,f'{param_display(n)} sudden {"rise" if
buf[-1]>buf[-2] else "drop"}'))
```

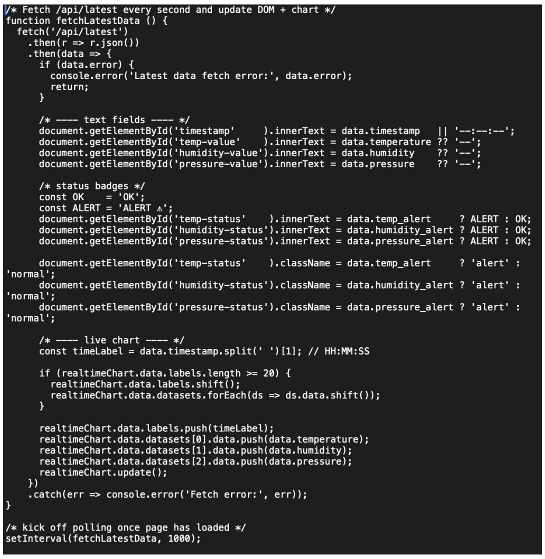
```
# ---------- DB init ----------
def init_db():
    conn = sqlite3.connect('environment.db'); cur = conn.cursor()
    cur.execute("""CREATE TABLE IF NOT EXISTS data (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    timestamp TEXT,
                    temperature REAL, humidity REAL, pressure REAL)""")
    cur.execute("""CREATE TABLE IF NOT EXISTS thresholds (
                    param TEXT PRIMARY KEY, min REAL, max REAL)""")
    cur.execute("""CREATE TABLE IF NOT EXISTS alerts (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    timestamp TEXT, type TEXT,
                    param TEXT, value REAL, message TEXT)""")
    if cur.execute("SELECT COUNT(*) FROM thresholds").fetchone()[0] == 0:
        for p in threshold_values:
            cur.execute("INSERT INTO thresholds VALUES (?,?,?)",
                        (p, threshold_values[p]['min'], threshold_values[p]['max']))
    else:
        for p, mn, mx in cur.execute("SELECT param,min,max FROM thresholds"):
            threshold_values[p]['min'] = float(mn); threshold_values[p]['max'] = float(mx)
    conn.commit(); conn.close()
```



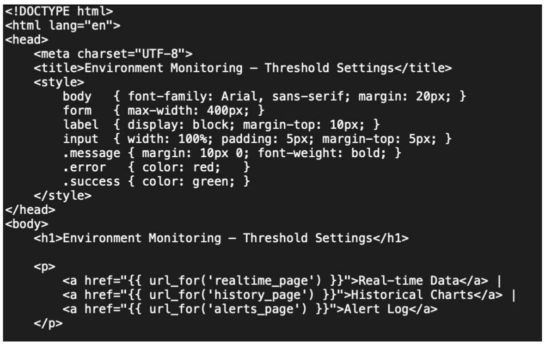
"environment.db" (144.0 KB) SQLite3 数据库

可用空间：21.3 GB(共：28.7 GB)

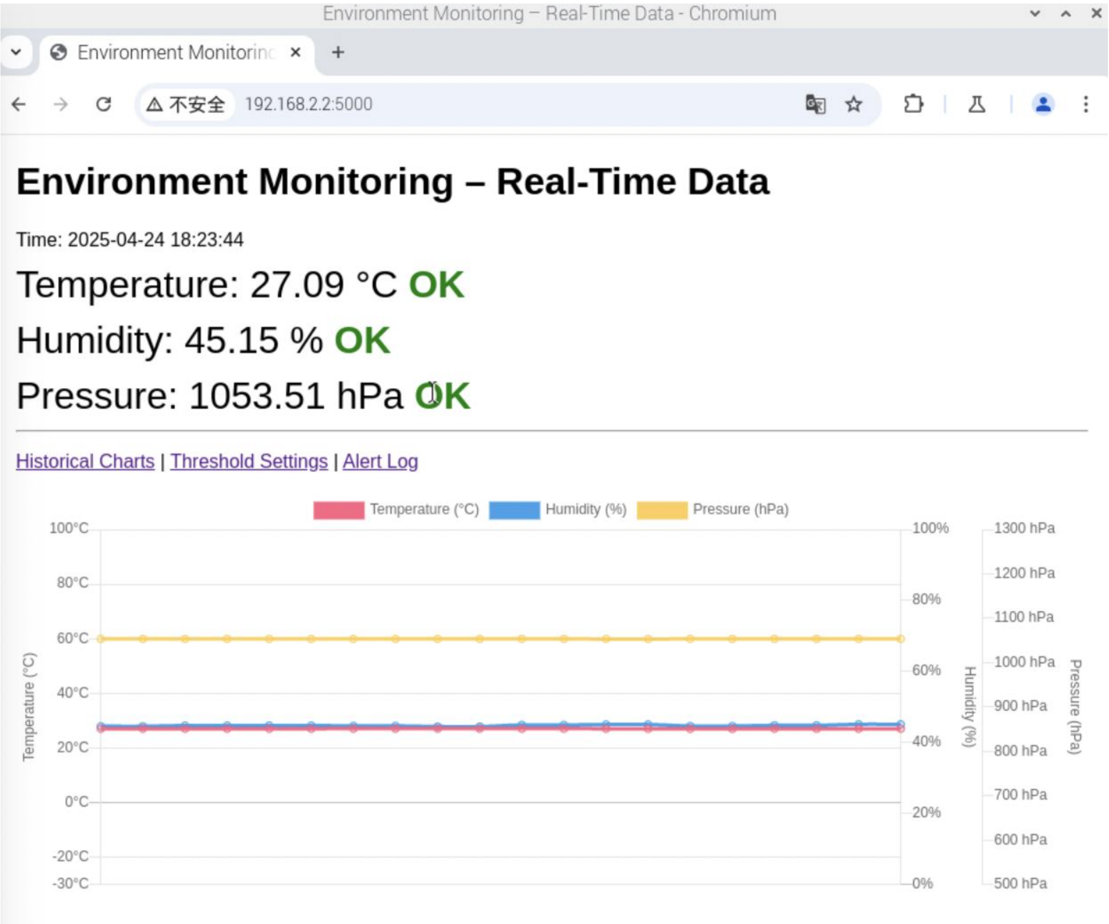
4 Web Interface

Item	How achieved
Responsive UI (1)	Simple fluid CSS; pages render on mobile & desktop.
Real-time updates (3)	static/script.js polls /api/latest every 1 s and updates DOM + Chart.js without page refresh. 
Customisable thresholds (4)	threshold.html form ➜ threshold_page(): validates, swaps min/max, writes to

```
/* Fetch /api/latest every second and update DOM + chart */
function fetchLatestData () {
  fetch('/api/latest')
    .then(r => r.json())
    .then(data => {
      if (data.error) {
        console.error('Latest data fetch error:', data.error);
        return;
      }

      /* ---- text fields ---- */
      document.getElementById('timestamp'     ).innerText = data.timestamp   || '--:--:--';
      document.getElementById('temp-value'    ).innerText = data.temperature ?? '--';
      document.getElementById('humidity-value').innerText = data.humidity    ?? '--';
      document.getElementById('pressure-value').innerText = data.pressure    ?? '--';

      /* status badges */
      const OK    = 'OK';
      const ALERT = 'ALERT ⚠';
      document.getElementById('temp-status'    ).innerText = data.temp_alert     ? ALERT : OK;
      document.getElementById('humidity-status').innerText = data.humidity_alert ? ALERT : OK;
      document.getElementById('pressure-status').innerText = data.pressure_alert ? ALERT : OK;

      document.getElementById('temp-status'    ).className = data.temp_alert     ? 'alert' :
'normal';
      document.getElementById('humidity-status').className = data.humidity_alert ? 'alert' :
'normal';
      document.getElementById('pressure-status').className = data.pressure_alert ? 'alert' :
'normal';

      /* ---- live chart ---- */
      const timeLabel = data.timestamp.split(' ')[1]; // HH:MM:SS

      if (realtimeChart.data.labels.length >= 20) {
        realtimeChart.data.labels.shift();
        realtimeChart.data.datasets.forEach(ds => ds.data.shift());
      }

      realtimeChart.data.labels.push(timeLabel);
      realtimeChart.data.datasets[0].data.push(data.temperature);
      realtimeChart.data.datasets[1].data.push(data.humidity);
      realtimeChart.data.datasets[2].data.push(data.pressure);
      realtimeChart.update();
    })
    .catch(err => console.error('Fetch error:', err));
}

/* kick off polling once page has loaded */
setInterval(fetchLatestData, 1000);
```

<table>
<tr><td></td><td>DB,

<pre>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Environment Monitoring – Threshold Settings</title>
    <style>
        body   { font-family: Arial, sans-serif; margin: 20px; }
        form   { max-width: 400px; }
        label { display: block; margin-top: 10px; }
        input { width: 100%; padding: 5px; margin-top: 5px; }
        .message { margin: 10px 0; font-weight: bold; }
        .error   { color: red;   }
        .success { color: green; }
    </style>
</head>
<body>
    <h1>Environment Monitoring – Threshold Settings</h1>

    <p>
        <a href="{{ url_for('realtime_page') }}">Real-time Data</a> |
        <a href="{{ url_for('history_page') }}">Historical Charts</a> |
        <a href="{{ url_for('alerts_page') }}">Alert Log</a>
    </p>
</pre>
updates in-memory limits.</td></tr>
<tr><td>Warning display (2)</td><td>Realtime page shows green OK or red ALERT badges; alerts.html colour-codes rows by alert type.</td></tr>
</table>



Environment Monitoring – Real-Time Data

Time: 2025-04-24 18:23:44

Temperature: 27.09 °C OK

Humidity: 45.15 % OK

Pressure: 1053.51 hPa OK

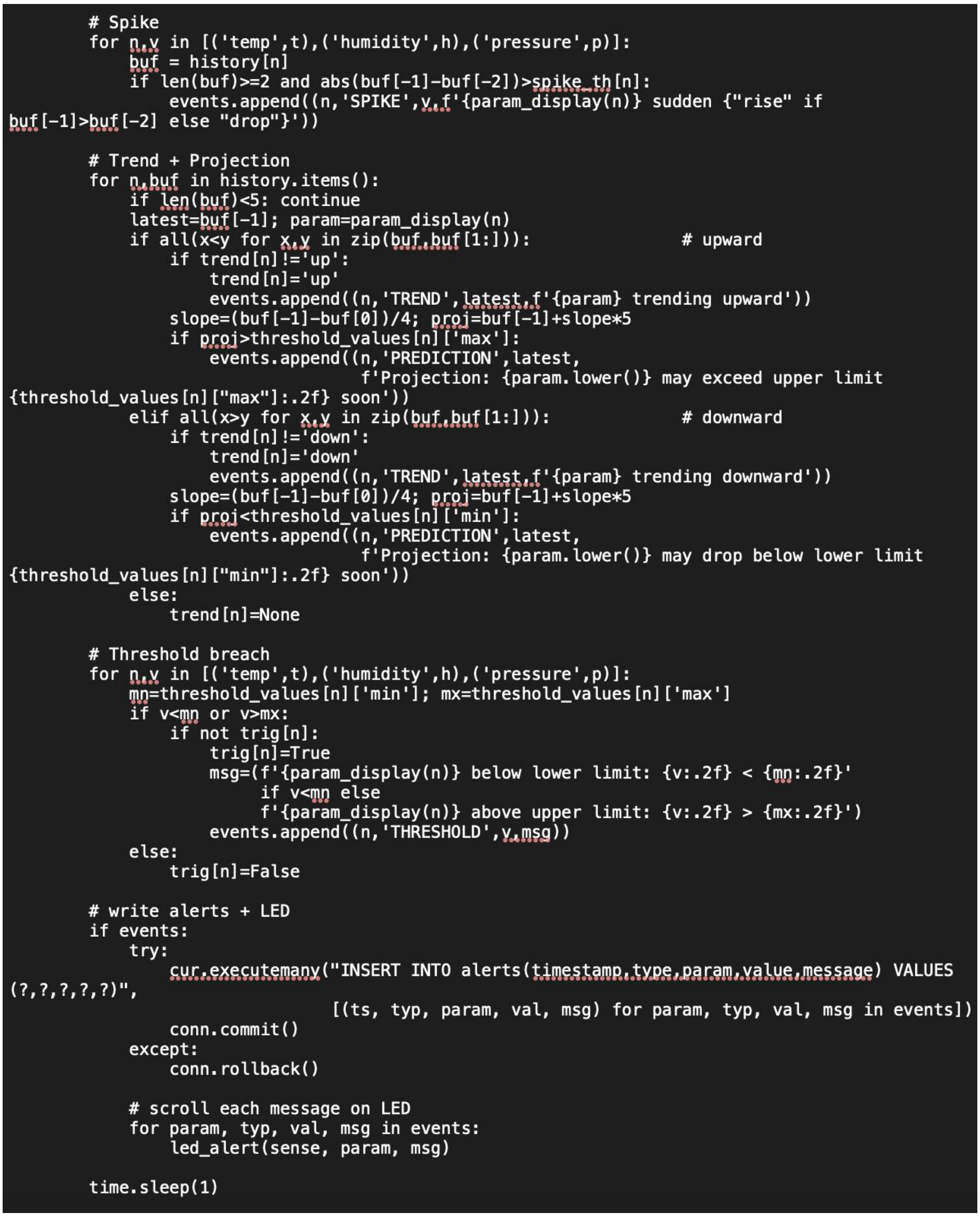
Historical Charts | Threshold Settings | Alert Log

5 Data Analysis & Reporting

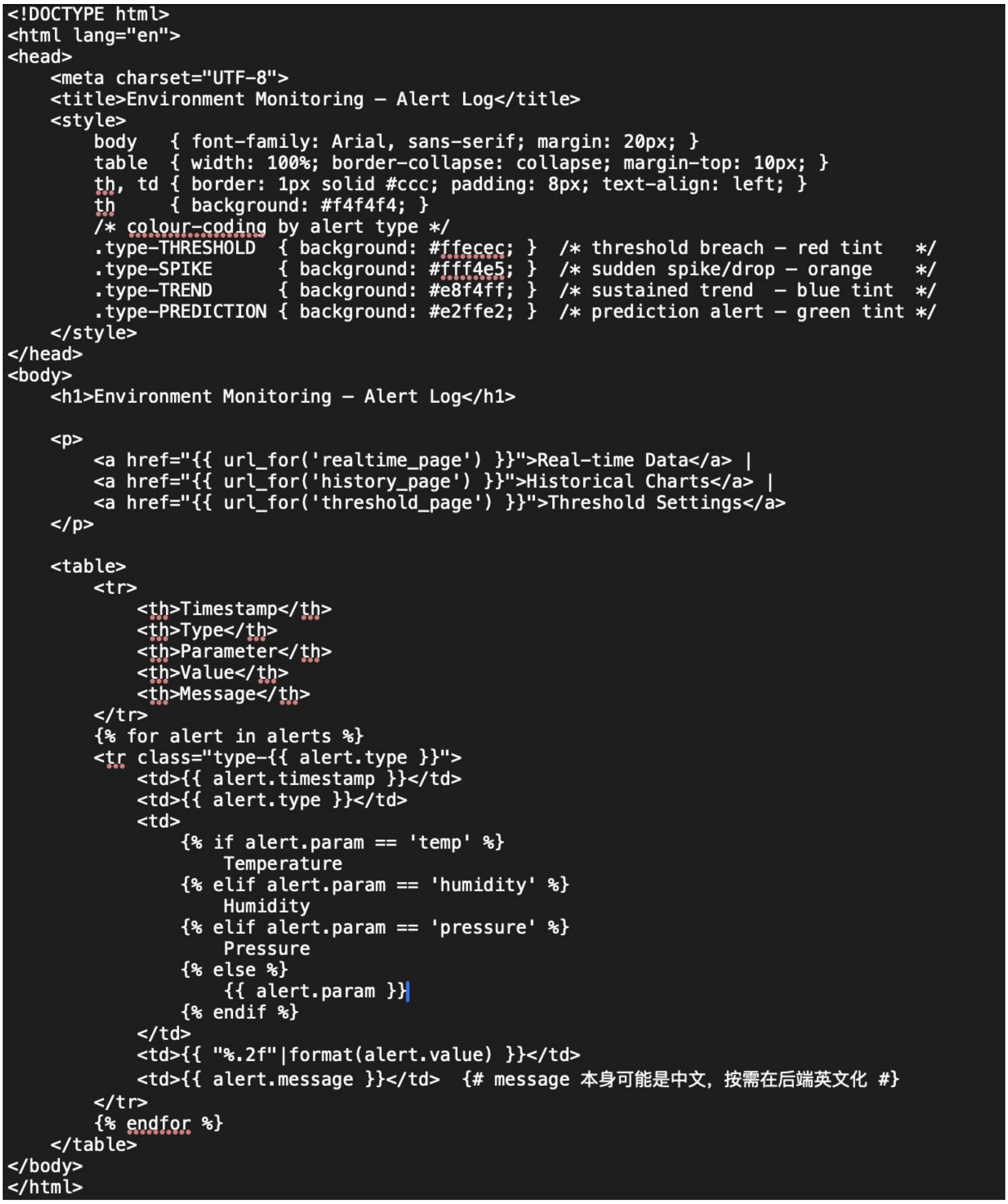
Function	Code lines & logic
Sudden spike / drop	Lines 96–103: abs(Δ) > spike_th → SPIKE alert.
Sustained trend	Lines 105–129: 5-point monotonic

	window → TREND alert.
Threshold prediction	Linear projection 5 steps ahead; cross-limit → PREDICTION alert.

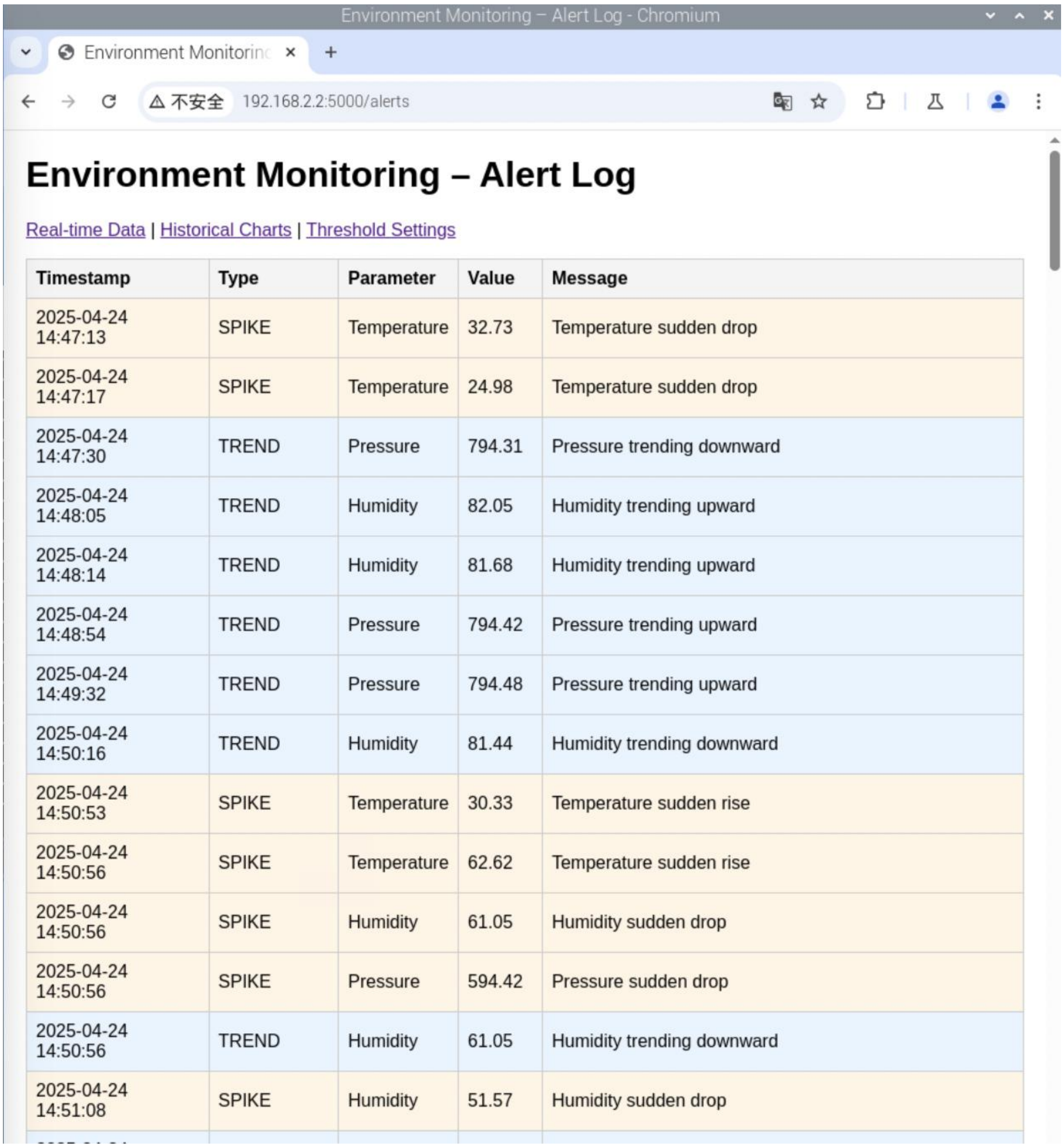
Alerts saved to alerts table and listed in alerts.html. Each new alert scrolls on LED matrix via led_alert().



```
        # Spike
        for n,v in [('temp',t),('humidity',h),('pressure',p)]:
            buf = history[n]
            if len(buf)>=2 and abs(buf[-1]-buf[-2])>spike_th[n]:
                events.append((n,'SPIKE',v,f'{param_display(n)} sudden {"rise" if 
buf[-1]>buf[-2] else "drop"}'))

        # Trend + Projection
        for n,buf in history.items():
            if len(buf)<5: continue
            latest=buf[-1]; param=param_display(n)
            if all(x<y for x,y in zip(buf,buf[1:])):               # upward
                if trend[n]!='up':
                    trend[n]='up'
                    events.append((n,'TREND',latest,f'{param} trending upward'))
                slope=(buf[-1]-buf[0])/4; proj=buf[-1]+slope*5
                if proj>threshold_values[n]['max']:
                    events.append((n,'PREDICTION',latest,
                                   f'Projection: {param.lower()} may exceed upper limit 
{threshold_values[n]["max"]:.2f} soon'))
            elif all(x>y for x,y in zip(buf,buf[1:])):             # downward
                if trend[n]!='down':
                    trend[n]='down'
                    events.append((n,'TREND',latest,f'{param} trending downward'))
                slope=(buf[-1]-buf[0])/4; proj=buf[-1]+slope*5
                if proj<threshold_values[n]['min']:
                    events.append((n,'PREDICTION',latest,
                                   f'Projection: {param.lower()} may drop below lower limit 
{threshold_values[n]["min"]:.2f} soon'))
            else:
                trend[n]=None

        # Threshold breach
        for n,v in [('temp',t),('humidity',h),('pressure',p)]:
            mn=threshold_values[n]['min']; mx=threshold_values[n]['max']
            if v<mn or v>mx:
                if not trig[n]:
                    trig[n]=True
                    msg=(f'{param_display(n)} below lower limit: {v:.2f} < {mn:.2f}'
                         if v<mn else
                         f'{param_display(n)} above upper limit: {v:.2f} > {mx:.2f}')
                    events.append((n,'THRESHOLD',v,msg))
            else:
                trig[n]=False

        # write alerts + LED
        if events:
            try:
                cur.executemany("INSERT INTO alerts(timestamp,type,param,value,message) VALUES 
(?,?,?,?,?)",
                                [(ts, typ, param, val, msg) for param, typ, val, msg in events])
                conn.commit()
            except:
                conn.rollback()

            # scroll each message on LED
            for param, typ, val, msg in events:
                led_alert(sense, param, msg)

        time.sleep(1)
```



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Environment Monitoring - Alert Log</title>
    <style>
        body   { font-family: Arial, sans-serif; margin: 20px; }
        table  { width: 100%; border-collapse: collapse; margin-top: 10px; }
        th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }
        th     { background: #f4f4f4; }
        /* colour-coding by alert type */
        .type-THRESHOLD  { background: #ffecec; }  /* threshold breach - red tint   */
        .type-SPIKE      { background: #fff4e5; }  /* sudden spike/drop - orange    */
        .type-TREND      { background: #e8f4ff; }  /* sustained trend  - blue tint */
        .type-PREDICTION { background: #e2ffe2; }  /* prediction alert - green tint */
    </style>
</head>
<body>
    <h1>Environment Monitoring - Alert Log</h1>

    <p>
        <a href="{{ url_for('realtime_page') }}">Real-time Data</a> |
        <a href="{{ url_for('history_page') }}">Historical Charts</a> |
        <a href="{{ url_for('threshold_page') }}">Threshold Settings</a>
    </p>

    <table>
        <tr>
            <th>Timestamp</th>
            <th>Type</th>
            <th>Parameter</th>
            <th>Value</th>
            <th>Message</th>
        </tr>
        {% for alert in alerts %}
        <tr class="type-{{ alert.type }}">
            <td>{{ alert.timestamp }}</td>
            <td>{{ alert.type }}</td>
            <td>
                {% if alert.param == 'temp' %}
                    Temperature
                {% elif alert.param == 'humidity' %}
                    Humidity
                {% elif alert.param == 'pressure' %}
                    Pressure
                {% else %}
                    {{ alert.param }}
                {% endif %}
            </td>
            <td>{{ "%.2f"|format(alert.value) }}</td>
            <td>{{ alert.message }}</td>  {# message 本身可能是中文，按需在后端英文化 #}
        </tr>
        {% endfor %}
    </table>
</body>
</html>
```



Environment Monitoring – Alert Log

Real-time Data | Historical Charts | Threshold Settings

Timestamp	Type	Parameter	Value	Message
2025-04-24 14:47:13	SPIKE	Temperature	32.73	Temperature sudden drop
2025-04-24 14:47:17	SPIKE	Temperature	24.98	Temperature sudden drop
2025-04-24 14:47:30	TREND	Pressure	794.31	Pressure trending downward
2025-04-24 14:48:05	TREND	Humidity	82.05	Humidity trending upward
2025-04-24 14:48:14	TREND	Humidity	81.68	Humidity trending upward
2025-04-24 14:48:54	TREND	Pressure	794.42	Pressure trending upward
2025-04-24 14:49:32	TREND	Pressure	794.48	Pressure trending upward
2025-04-24 14:50:16	TREND	Humidity	81.44	Humidity trending downward
2025-04-24 14:50:53	SPIKE	Temperature	30.33	Temperature sudden rise
2025-04-24 14:50:56	SPIKE	Temperature	62.62	Temperature sudden rise
2025-04-24 14:50:56	SPIKE	Humidity	61.05	Humidity sudden drop
2025-04-24 14:50:56	SPIKE	Pressure	594.42	Pressure sudden drop
2025-04-24 14:50:56	TREND	Humidity	61.05	Humidity trending downward
2025-04-24 14:51:08	SPIKE	Humidity	51.57	Humidity sudden drop

6 System Reliability & Error Handling

All DB writes in try/except; on failure rollback() and loop continues.
SQLite opened with check_same_thread=False for safe crossthread use.
Invalid threshold input returns friendly error message.
If Sense HAT library unavailable, thread sleeps and retries → service keeps running.

7 Code Quality & Maintainability

Follows PEP8 names; major functions have docstrings.
SQL uses placeholders – no injection risk.
Repeated logic handled via small helper loops; no magic numbers.
Files grouped into templates/ and static/ to match Flask convention.

8 Security Analysis & Measures

Risk	Current status	Planned mitigation
Unauthorised access	Dashboard open on LAN	
CSRF	Not enabled	Integrate flask_wtf.CSRFProtect()
Rate abuse	Unlimited API calls	Add flask_limiter (e.g. 60 req/min/IP)
Plain-text DB	Local only	Use SQLCipher or move to cloud DB with IAM
Error info leak	Debug off but traceback possible	Custom error pages, logging only to file

9 Setup & Configuration

1. Install SenseEMU

```
sudo apt update
sudo apt install python3-sense-emu sense-emu-tools -y
```

2. Clone project

```
mkdir ~/lee && cd ~/lee
```

3. Install Flask

```
sudo apt install python3-flask -y
```

4. Run

```
python3 app.py
# browse to http://192.168.2.2:5000
```

5. Stop – Ctrl+C in terminal.

10 User Manual

Realtime Data – opens at /lee/; updates every second; red ALERT badge signals outofrange value.
Historical Charts – /lee/templates/history; view longterm trends.
Threshold Settings – /lee/templates/threshold; edit limits then Save.
Alert Log – /lee/templates/alerts; colourcoded list of all alerts.
LED Matrix – scrolls first 8 chars of alert message in colour (T = red, H = blue, P = green).

11 Troubleshooting Guide

Symptom	Cause	Fix
Page not loading	Flask not running / wrong IP	Confirm "Running on …:5000" in terminal; check Pi IP (hostname -I).
Dashboard shows "No data"	Sense-EMU not installed	sudo apt install python3-sense-emu; run sense_emu_gui.
Repeated DB errors	DB locked	Stop app, rename environment.db, restart (new DB auto-created).
LED text unreadable	Brightness too high	Emulator: tick *Low-light*; hardware: sense.low_light=True is set.
Wrong time zone	System clock	sudo raspi-config → Localisation → Timezone, reboot.

12 Conclusion

The system fulfils all required functionality: accurate data capture and storage, responsive web interface with live charts, automatic alerts with LED feedback, robust error handling, and clear, maintainable code. Future iterations will implement the documented security measures, push data to cloud storage, and add new sensors for an even richer environmental profile.