08 - RouteTrack Pi — Local Web Dashboard (Flask API + Leaflet Map)

Date: December 25, 2025
Category: Raspberry Pi / GPS / Flask / Leaflet / Dashboard

Backlink: 07 – RouteTrack Pi — Automated Route Processing (systemd Service + Timer)

Project Goal

This phase creates a **local web dashboard** hosted on the Pi that:

- Shows the recorded route on a map (Leaflet)
- Displays stop markers
- Shows daily summary stats
- Reads from SQLite only (safe, no DB lock risk)

RouteTrack now becomes usable *in real time* via a browser on the local network.

Dashboard Architecture

Component	Purpose
Flask app	Serves API + webpage
SQLite	Data source (`gps_points`, `stop_events`, `daily_summary`)

Component	Purpose
Leaflet	Map rendering (browser)
OpenStreetMap tiles	Basemap tiles

Install Dashboard Dependencies (venv)

You already confirmed Flask is installed in the venv. For Leaflet, we don’t need a Python package — it’s loaded in the browser.

If you want date parsing helpers later, we can add them, but for now **keep it minimal**.

(You already did these earlier, included here for completeness.)

```
/opt/routetrack/venv/bin/pip install --upgrade pip
/opt/routetrack/venv/bin/pip install flask gunicorn
```

Sanity check:

```
/opt/routetrack/venv/bin/python -c "import flask; print('Flask OK')"
```

Create Flask App Folder

```
sudo mkdir -p /opt/routetrack/web/templates /opt/routetrack/web/static
sudo chown -R $USER:$USER /opt/routetrack/web
```

Create the Flask API App

Create:

```
sudo nano /opt/routetrack/web/app.py
```

Paste:

```
#!/usr/bin/env python3
"""
RouteTrack Local Dashboard (Flask)
----------------------------------

Provides:
- Web UI page (Leaflet map)
- JSON API endpoints:
  - /api/summary/<date>
  - /api/points/<date>
  - /api/stops/<date>

Notes:
- This dashboard is READ-ONLY.
- It never writes to SQLite (avoids lock contention).
"""

import sqlite3
from flask import Flask, jsonify, render_template

DB_PATH = "/opt/routetrack/data/routetrack.sqlite"

app = Flask(__name__)

def db():
    conn = sqlite3.connect(DB_PATH)
    conn.row_factory = sqlite3.Row
    return conn

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/api/summary/<day>")
def api_summary(day):
    conn = db()
    cur = conn.cursor()
    cur.execute("SELECT * FROM daily_summary WHERE date = ?", (day,))
```

```
    row = cur.fetchone()
    conn.close()

    if not row:
        return jsonify({"error": "No summary for this date"}), 404

    return jsonify(dict(row))

@app.route("/api/points/<day>")
def api_points(day):
    conn = db()
    cur = conn.cursor()

    start = f"{day}T00:00:00Z"
    end = f"{day}T23:59:59Z"

    cur.execute("""
        SELECT ts, lat, lon
        FROM gps_points
        WHERE ts >= ? AND ts <= ?
          AND mode = 3
          AND lat IS NOT NULL
          AND lon IS NOT NULL
        ORDER BY ts
    """, (start, end))

    rows = cur.fetchall()
    conn.close()

    # Return as list of [lat, lon]
    points = [[r["lat"], r["lon"]] for r in rows]
    return jsonify(points)

@app.route("/api/stops/<day>")
def api_stops(day):
    conn = db()
    cur = conn.cursor()

    start = f"{day}T00:00:00Z"
    end = f"{day}T23:59:59Z"
```

```
    cur.execute("""
        SELECT start_ts, end_ts, duration_seconds, lat, lon
        FROM stop_events
        WHERE start_ts >= ? AND start_ts <= ?
        ORDER BY start_ts
    """, (start, end))

    rows = cur.fetchall()
    conn.close()

    stops = [dict(r) for r in rows]
    return jsonify(stops)

if __name__ == "__main__":
    # Local dev run
    app.run(host="0.0.0.0", port=5000, debug=False)
```

Make executable:

```
sudo chmod +x /opt/routetrack/web/app.py
```

Create the Leaflet Web Page

Create:

```
sudo nano /opt/routetrack/web/templates/index.html
```

Paste:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>RouteTrack Dashboard</title>
  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <!-- Leaflet (CDN) -->
```

```
  <link
    rel="stylesheet"
    href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
  />
  <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>

  <style>
    body { margin: 0; font-family: Arial, sans-serif; }
    #topbar { padding: 10px; background: #111; color: #fff; }
    #map { height: 70vh; }
    #stats { padding: 10px; }
    .row { margin: 6px 0; }
    code { background: #eee; padding: 2px 4px; border-radius: 4px; }
  </style>
</head>

<body>
  <div id="topbar">
    <strong>RouteTrack</strong> — Local Dashboard
      |  
    Date: <input id="day" type="date" />
    <button onclick="loadAll()">Load</button>
  </div>

  <div id="map"></div>
  <div id="stats">
    <h3>Daily Summary</h3>
    <div id="summary"></div>
    <h3>Stops</h3>
    <div id="stops"></div>
  </div>

<script>
  // Default date = today (browser local time)
  const dayInput = document.getElementById("day");
  dayInput.valueAsDate = new Date();

  const map = L.map("map").setView([38.7153, -89.94], 13);

  L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
```

```
  maxZoom: 19,
  attribution: "&copy; OpenStreetMap contributors"
}).addTo(map);

let routeLine = null;
let stopMarkers = [];

async function loadAll() {
  const day = dayInput.value;
  await loadRoute(day);
  await loadStops(day);
  await loadSummary(day);
}

async function loadRoute(day) {
  const res = await fetch(`/api/points/${day}`);
  const pts = await res.json();

  if (routeLine) map.removeLayer(routeLine);
  if (!pts.length) return;

  routeLine = L.polyline(pts, { weight: 4 }).addTo(map);
  map.fitBounds(routeLine.getBounds());
}

async function loadStops(day) {
  // clear old markers
  stopMarkers.forEach(m => map.removeLayer(m));
  stopMarkers = [];

  const res = await fetch(`/api/stops/${day}`);
  const stops = await res.json();

  const stopsDiv = document.getElementById("stops");
  stopsDiv.innerHTML = "";

  if (!Array.isArray(stops) || !stops.length) {
    stopsDiv.innerHTML = "<div class='row'>No stops found.</div>";
    return;
  }
```

```
    stops.forEach(s => {
      const durMin = Math.round(s.duration_seconds / 60);
      stopsDiv.innerHTML += `<div class="row">
        Stop: <code>${s.start_ts}</code> → <code>${s.end_ts}</code>
        (${durMin} min)
      </div>`;

      if (s.lat && s.lon) {
        const m = L.marker([s.lat, s.lon]).addTo(map)
          .bindPopup(`Stop (${durMin} min)<br>${s.start_ts}`);
        stopMarkers.push(m);
      }
    });
  }

  async function loadSummary(day) {
    const summaryDiv = document.getElementById("summary");
    summaryDiv.innerHTML = "";

    const res = await fetch(`/api/summary/${day}`);
    const data = await res.json();

    if (data.error) {
      summaryDiv.innerHTML = `<div class="row">No summary for ${day}. Run processor first.</div>`;
      return;
    }

    summaryDiv.innerHTML = `
      <div class="row">Start: <code>${data.start_ts}</code></div>
      <div class="row">End: <code>${data.end_ts}</code></div>
      <div class="row">Distance: <strong>${data.total_distance_miles}</strong> miles</div>
      <div class="row">Moving: <strong>${Math.round(data.moving_time_seconds/60)}</strong> minutes</div
>
      <div class="row">Stopped: <strong>${Math.round(data.stopped_time_seconds/60)}</strong> minutes</
div>
      <div class="row">Stops: <strong>${data.stop_count}</strong></div>
    `;
  }
```

```
  // Auto-load on page open
  loadAll();
</script>
</body>
</html>
```

Run the Dashboard (Manual Test)

```
/opt/routetrack/venv/bin/python /opt/routetrack/web/app.py
```

Then browse from your LAN:

- `http://<PI-IP>:5000`

Find your Pi IP:

```
hostname -I
```

Stop the server with `Ctrl+C`.

Next Step (After Manual Test)

Next phase is productionizing the dashboard:

- systemd service for Flask (Gunicorn)
- optional Nginx reverse proxy
- optional local authentication
- optional “live view” tracking

Revision #2
Created 25 December 2025 15:52:24 by Nate Nash
Updated 25 December 2025 21:16:37 by Nate Nash