12 - RouteTrack Pi — Mobile UX Upgrade (Shift Controls + Status)

Date: December 25th, 2025
Category: Raspberry Pi / GPS / Web UI / UX

Backlink: 08 — RouteTrack Pi — Local Dashboard (Leaflet + Flask)

Project Goal

This update makes the RouteTrack dashboard **easy and safe to use from a phone**, especially since this Pi will be:

- powered down frequently
- used on the go
- accessed quickly before/after driving

The dashboard becomes a **real operator UI**:

- you can glance and instantly know whether a shift is active
- you can start/stop with big buttons (no scrolling)
- the UI recovers cleanly after reboot

What We Added (UX Features)

1) Shift Status Badge (Top Bar)

Shows:

- **ACTIVE** (with start time)
- **STOPPED**

2) Sticky Shift Controls (Bottom Bar)

Large buttons designed for mobile thumbs:

- Start Shift (green)
- Stop Shift (red)

3) Button Safety Rules

- If shift is ACTIVE → Start disabled
- If shift is STOPPED → Stop disabled

4) Inline Toast Messages

Non-blocking confirmations like:

- “Shift started”
- “Shift ended”
- “API not reachable yet” (during boot)

5) Auto Refresh After Start/Stop

After changing shift state, the UI automatically refreshes:

- route line
- stops
- daily summary
- shift card

File Updates

Update `app.py` (Shift endpoints + existing APIs)

Replace the current `/opt/routetrack/web/app.py` with this full version.

```
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 (read-only route data):
  - /api/summary/<date>
  - /api/points/<date>
  - /api/stops/<date>

Shift Control Endpoints (write minimal shift state only):
  - GET  /api/shift/active
  - POST /api/shift/start
  - POST /api/shift/stop
  - GET  /api/shift/summary

Notes:
- Route endpoints are READ-ONLY from gps_points/stop_events/daily_summary.
- Shift endpoints write only to the shifts table.
"""

import sqlite3
from datetime import datetime, timezone
from flask import Flask, jsonify, render_template, request

DB_PATH = "/opt/routetrack/data/routetrack.sqlite"
app = Flask(__name__)

def db():
    conn = sqlite3.connect(DB_PATH, timeout=10)
    conn.row_factory = sqlite3.Row
    return conn

def utc_now_iso():
    # ISO-8601 with Z suffix (matches gps_points ts style)
    return datetime.now(timezone.utc).replace(microsecond=0).isoformat().replace("+00:00", "Z")

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

# -------------------------
```

```
# Health (optional)
# -------------------------
@app.route("/api/health")
def api_health():
    try:
        conn = db()
        conn.execute("SELECT 1;")
        conn.close()
        return jsonify({"ok": True})
    except Exception as e:
        return jsonify({"ok": False, "error": str(e)}), 500

# -------------------------
# Route data endpoints
# -------------------------
@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()

    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()

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

# -------------------------
# Shift endpoints
# -------------------------
@app.route("/api/shift/active")
def api_shift_active():
    conn = db()
    cur = conn.cursor()
    cur.execute("""
```

```
        SELECT id, start_ts, end_ts
        FROM shifts
        WHERE end_ts IS NULL
        ORDER BY id DESC
        LIMIT 1
    """)
    row = cur.fetchone()
    conn.close()

    if not row:
        return jsonify({"active": False})

    return jsonify({
        "active": True,
        "id": row["id"],
        "start_ts": row["start_ts"],
        "end_ts": row["end_ts"],
    })

@app.route("/api/shift/start", methods=["POST"])
def api_shift_start():
    # If already active, do nothing (idempotent-ish)
    conn = db()
    cur = conn.cursor()

    cur.execute("SELECT id, start_ts FROM shifts WHERE end_ts IS NULL ORDER BY id DESC LIMIT 1;")
    existing = cur.fetchone()
    if existing:
        conn.close()
        return jsonify({"ok": True, "message": "Shift already active", "id": existing["id"], "start_ts": existing[
"start_ts"]})

    start_ts = utc_now_iso()
    cur.execute("INSERT INTO shifts (start_ts) VALUES (?);", (start_ts,))
    conn.commit()

    cur.execute("SELECT id, start_ts FROM shifts WHERE end_ts IS NULL ORDER BY id DESC LIMIT 1;")
    row = cur.fetchone()
    conn.close()
```

```
    return jsonify({"ok": True, "message": "Shift started", "id": row["id"], "start_ts": row["start_ts"]})

@app.route("/api/shift/stop", methods=["POST"])
def api_shift_stop():
    conn = db()
    cur = conn.cursor()

    cur.execute("SELECT id, start_ts FROM shifts WHERE end_ts IS NULL ORDER BY id DESC LIMIT 1;")
    row = cur.fetchone()
    if not row:
        conn.close()
        return jsonify({"ok": False, "error": "No active shift."}), 400

    end_ts = utc_now_iso()
    cur.execute("UPDATE shifts SET end_ts = ? WHERE id = ?;", (end_ts, row["id"]))
    conn.commit()
    conn.close()

    return jsonify({"ok": True, "message": "Shift ended", "id": row["id"], "start_ts": row["start_ts"], "end_ts":
end_ts})

@app.route("/api/shift/summary")
def api_shift_summary():
    conn = db()
    cur = conn.cursor()

    cur.execute("""
        SELECT id, start_ts, end_ts
        FROM shifts
        WHERE end_ts IS NULL
        ORDER BY id DESC
        LIMIT 1
    """)
    shift = cur.fetchone()
    conn.close()

    if not shift:
        return jsonify({"error": "No active shift."}), 404

    # Simple summary for the UI (elapsed seconds)
```

```
    start_dt = datetime.fromisoformat(shift["start_ts"].replace("Z", "+00:00"))
    now_dt = datetime.now(timezone.utc)
    elapsed_s = int((now_dt - start_dt).total_seconds())

    return jsonify({
        "active": True,
        "id": shift["id"],
        "start_ts": shift["start_ts"],
        "elapsed_seconds": elapsed_s
    })

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

Make executable (optional, harmless):

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

Restart dashboard:

```
sudo systemctl restart routetrack-dashboard.service
```

Quick verify:

```
curl http://localhost:5000/api/shift/active
```

Update `index.html` (Mobile UI + sticky controls + status)

Edit:

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

Paste the full file:

```
<!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>
    :root {
      --bg: #0f0f0f;
      --panel: #151515;
      --text: #f2f2f2;
      --muted: #bdbdbd;
      --ok: #16a34a;
      --stop: #dc2626;
      --warn: #f59e0b;
      --card: #1c1c1c;
      --border: #2b2b2b;
    }

    body { margin: 0; font-family: Arial, sans-serif; background: var(--bg); color: var(--text); }
    #topbar {
      padding: 10px 12px;
      background: var(--panel);
      color: var(--text);
      display: flex;
      gap: 10px;
      align-items: center;
      flex-wrap: wrap;
      border-bottom: 1px solid var(--border);
    }

    #brand { font-weight: 700; }
    #statusBadge {
      padding: 4px 10px;
      border-radius: 999px;
      font-size: 12px;
      border: 1px solid var(--border);
```

```
  background: var(--card);
}
.badge-active { border-color: rgba(22,163,74,.6); }
.badge-stopped { border-color: rgba(220,38,38,.6); }

#topControls {
  margin-left: auto;
  display: flex;
  gap: 8px;
  align-items: center;
}

input[type="date"]{
  background: var(--card);
  color: var(--text);
  border: 1px solid var(--border);
  border-radius: 8px;
  padding: 6px 8px;
}

button {
  border: 0;
  padding: 9px 12px;
  border-radius: 10px;
  font-weight: 700;
  cursor: pointer;
}
button:disabled { opacity: 0.45; cursor: not-allowed; }

.btn { background: #2a2a2a; color: var(--text); border: 1px solid var(--border); }
.btnStart { background: var(--ok); color: #fff; }
.btnStop { background: var(--stop); color: #fff; }

#map { height: 62vh; }

#content {
  padding: 12px;
  display: grid;
  gap: 12px;
}
```

```
.card {
  background: var(--card);
  border: 1px solid var(--border);
  border-radius: 14px;
  padding: 12px;
}

h3 { margin: 0 0 8px 0; }
.row { margin: 6px 0; color: var(--muted); }
code { background: #232323; padding: 2px 6px; border-radius: 6px; color: #fff; }

/* Sticky bottom control bar for mobile */
#shiftBar {
  position: sticky;
  bottom: 0;
  background: rgba(15,15,15,.92);
  backdrop-filter: blur(8px);
  border-top: 1px solid var(--border);
  padding: 10px 12px;
  display: flex;
  gap: 10px;
  z-index: 999;
}
#shiftBar button {
  flex: 1;
  padding: 14px 12px;
  border-radius: 14px;
  font-size: 16px;
}

/* Toast */
#toast {
  position: fixed;
  left: 50%;
  transform: translateX(-50%);
  bottom: 86px;
  background: #111;
  border: 1px solid var(--border);
  color: var(--text);
```

```
    padding: 10px 12px;
    border-radius: 12px;
    display: none;
    z-index: 1000;
    max-width: 92vw;
   }
   #toast.ok { border-color: rgba(22,163,74,.7); }
   #toast.err { border-color: rgba(220,38,38,.7); }
   #toast.warn { border-color: rgba(245,158,11,.7); }

   @media (min-width: 900px) {
    #map { height: 70vh; }
    #shiftBar { width: 520px; margin: 0 auto 12px auto; border-radius: 14px; }
   }
  </style>
</head>

<body>
  <div id="topbar">
   <span id="brand">RouteTrack</span>
   <span id="statusBadge" class="badge-stopped">□□ SHIFT STOPPED</span>

   <div id="topControls">
    <span style="color: var(--muted); font-size: 12px;">Date</span>
    <input id="day" type="date" />
    <button class="btn" onclick="loadAll()">Reload</button>
   </div>
  </div>

  <div id="map"></div>

  <div id="content">
   <div class="card">
    <h3>Active Shift</h3>
    <div id="shiftCard" class="row">Checking shift status…</div>
   </div>

   <div class="card">
    <h3>Daily Summary</h3>
    <div id="summary" class="row">Loading…</div>
```

```
    </div>

    <div class="card">
      <h3>Stops</h3>
      <div id="stops" class="row">Loading…</div>
    </div>
  </div>

  <!-- Sticky mobile shift controls -->
  <div id="shiftBar">
    <button id="btnStart" class="btnStart" onclick="startShift()">Start Shift</button>
    <button id="btnStop" class="btnStop" onclick="stopShift()">Stop Shift</button>
  </div>

  <div id="toast"></div>

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

  const statusBadge = document.getElementById("statusBadge");
  const btnStart = document.getElementById("btnStart");
  const btnStop = document.getElementById("btnStop");
  const toastEl = document.getElementById("toast");

  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 = [];

  function toast(msg, type="ok") {
    toastEl.className = "";
    toastEl.classList.add(type);
    toastEl.textContent = msg;
    toastEl.style.display = "block";
```

```
  setTimeout(() => toastEl.style.display = "none", 3200);
}

function setShiftUI(active, start_ts=null) {
  if (active) {
    statusBadge.textContent = `□□ SHIFT ACTIVE${start_ts ? " — " + start_ts : ""}`;
    statusBadge.classList.remove("badge-stopped");
    statusBadge.classList.add("badge-active");
    btnStart.disabled = true;
    btnStop.disabled = false;
  } else {
    statusBadge.textContent = "□□ SHIFT STOPPED";
    statusBadge.classList.remove("badge-active");
    statusBadge.classList.add("badge-stopped");
    btnStart.disabled = false;
    btnStop.disabled = true;
  }
}

async function apiJSON(url, opts={}) {
  const res = await fetch(url, opts);
  let data = {};
  try { data = await res.json(); } catch(e) {}
  return { res, data };
}

async function refreshShiftState() {
  const { res, data } = await apiJSON("/api/shift/active");
  if (!res.ok) {
    setShiftUI(false);
    document.getElementById("shiftCard").textContent = "Shift API not reachable yet.";
    return;
  }

  setShiftUI(!!data.active, data.start_ts || null);

  if (data.active) {
    // Show elapsed time quickly
    const { data: sum } = await apiJSON("/api/shift/summary");
    if (!sum || sum.error) {
```

```
      document.getElementById("shiftCard").innerHTML =
        `<div class="row">Active shift detected. Start: <code>${data.start_ts}</code></div>`;
      return;
    }
    const mins = Math.floor((sum.elapsed_seconds || 0) / 60);
    document.getElementById("shiftCard").innerHTML =
      `<div class="row">Started: <code>${sum.start_ts}</code></div>
       <div class="row">Elapsed: <strong>${mins}</strong> min</div>`;
  } else {
    document.getElementById("shiftCard").textContent = "No active shift.";
  }
}

async function startShift() {
  btnStart.disabled = true;
  const { res, data } = await apiJSON("/api/shift/start", { method: "POST" });
  if (!res.ok || data.ok === false) {
    toast(data.error || "Failed to start shift.", "err");
    await refreshShiftState();
    return;
  }
  toast(data.message || "Shift started.", "ok");
  await refreshShiftState();
  await loadAll(); // refresh route/stops/summary
}

async function stopShift() {
  // One simple safety check (no popup spam):
  if (!confirm("End shift now?")) return;

  btnStop.disabled = true;
  const { res, data } = await apiJSON("/api/shift/stop", { method: "POST" });
  if (!res.ok || data.ok === false) {
    toast(data.error || "Failed to stop shift.", "err");
    await refreshShiftState();
    return;
  }
  toast(data.message || "Shift ended.", "ok");
  await refreshShiftState();
  await loadAll();
```

```
}

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

async function loadRoute(day) {
  const { res, data } = await apiJSON(`/api/points/${day}`);
  const pts = Array.isArray(data) ? data : [];

  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) {
  stopMarkers.forEach(m => map.removeLayer(m));
  stopMarkers = [];

  const { data } = await apiJSON(`/api/stops/${day}`);
  const stops = Array.isArray(data) ? data : [];

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

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

  stops.forEach(s => {
    const durMin = Math.round((s.duration_seconds || 0) / 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 { data } = await apiJSON(`/api/summary/${day}`);

    if (!data || 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>
    `;
  }

  // Boot behavior: shift state first, then route data
  (async () => {
    try {
      await refreshShiftState();
      await loadAll();
    } catch (e) {
      toast("Dashboard loading… waiting on services.", "warn");
```

```
    }

    // Light auto-refresh of shift status every 15s
    setInterval(refreshShiftState, 15000);
  })();
</script>
</body>
</html>
```

Restart dashboard:

```
sudo systemctl restart routetrack-dashboard.service
```

Verification

Confirm shift API works

```
curl http://localhost:5000/api/shift/active
```

Start shift:

```
curl -X POST http://localhost:5000/api/shift/start
```

Stop shift:

```
curl -X POST http://localhost:5000/api/shift/stop
```

Then load the dashboard from your phone and confirm:

- status badge flips correctly
- buttons enable/disable properly
- map + stats refresh after shift actions

Why This UX Matters (for a portable device)

This dashboard is now resilient for:

- frequent power-off/on cycles
- quick “start shift / drive / stop shift” workflows
- using the UI one-handed on a phone

It reduces mistakes and removes uncertainty — which is exactly what you want when this becomes a daily tool.

Next Steps

1. Add **“Shift view” mode** (show only points within the active shift window)
2. Add **start/stop shift button inside the map** (floating control)
3. Improve stop detection with:
   - ignition off detection (optional)
   - drift suppression using epx/epy thresholds

Revision #2
Created 25 December 2025 17:55:31 by Nate Nash
Updated 25 December 2025 21:17:06 by Nate Nash