10 - RouteTrack Pi — Shift Mode (SQLite + Flask API + Dashboard Controls)

Date: December 25, 2025
Category: Raspberry Pi / GPS / SQLite / Flask / Leaflet
Backlink: 09 – RouteTrack Pi — Dashboard Autostart (Gunicorn + systemd)

Project Goal

This phase introduces **Shift Mode** to RouteTrack.

Shift Mode allows RouteTrack to track work sessions independently of calendar days, which is essential for a portable, vehicle-mounted system that:

- Is powered down frequently
- Moves between locations
- Does not follow strict midnight-to-midnight boundaries
- Needs accurate per-shift metrics (hours, stops, time-on-site)

The dashboard now supports a simple workflow: **Start Shift → Drive → End Shift**

Why Shift Mode Matters

Daily summaries work well for reporting, but they don’t match real-world truck usage:

- Overnight work can cross calendar boundaries
- Reboots/power loss interrupt sessions
- Short test runs clutter daily totals

Shift Mode solves this by creating a clean “session boundary” that the user controls.

Database Changes

New `shifts` Table

A new SQLite table stores shift metadata independently of GPS data.

Table: `shifts`

```
CREATE TABLE IF NOT EXISTS shifts (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  start_ts TEXT NOT NULL,
  end_ts TEXT,
  note TEXT
);

CREATE INDEX IF NOT EXISTS idx_shifts_start_ts
  ON shifts(start_ts);
```

Design Notes:

- `start_ts` and `end_ts` stored as UTC ISO-8601 strings
- `end_ts` stays `NULL` while a shift is active
- Only one active shift allowed at a time
- Lightweight, isolated table to minimize lock contention

Applying the Schema Safely

Because GPS logging writes constantly to SQLite, stop the logger before applying schema changes.

```
sudo systemctl stop routetrack-logger.service
sqlite3 /opt/routetrack/data/routetrack.sqlite < /opt/routetrack/config/schema.sql
sqlite3 /opt/routetrack/data/routetrack.sqlite ".tables"
sudo systemctl start routetrack-logger.service
```

Expected tables:

```
daily_summary  gps_points  shifts  stop_events
```

Flask API Enhancements (Full `app.py`)

Shift Mode is implemented via additional Flask API endpoints.

New Endpoints

Method	Endpoint	Purpose
GET	`/api/shift/active`	Returns the active shift (if any)
POST	`/api/shift/start`	Starts a new shift
POST	`/api/shift/end`	Ends the active shift
GET	`/api/shift/summary`	Returns live stats for the active shift

Replace `/opt/routetrack/web/app.py`

Edit:

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

Paste the full file:

```
#!/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>

Shift Mode endpoints:
  - GET  /api/shift/active
  - POST /api/shift/start
  - POST /api/shift/end
  - GET  /api/shift/summary

Notes:
- This dashboard is READ-ONLY for GPS-derived tables:
    gps_points, stop_events, daily_summary
- Shift Mode DOES write to SQLite, but only into the "shifts" table.
  This avoids lock contention with the logger and keeps writes minimal.
"""

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():
    """
    Open SQLite connection with Row output so we can jsonify results
    via dict(row).
    """
    conn = sqlite3.connect(DB_PATH)
    conn.row_factory = sqlite3.Row
    return conn
```

```
def utc_now_iso():
    """
    Return current UTC timestamp in ISO-8601 format (no microseconds).
    Example: 2025-12-25T16:05:00+00:00
    """
    return datetime.now(timezone.utc).replace(microsecond=0).isoformat()


@app.route("/")
def index():
    """Serve the dashboard HTML page (Leaflet UI)."""
    return render_template("index.html")


# ============================================================
# Existing Daily Views (READ-ONLY)
# ============================================================

@app.route("/api/summary/<day>")
def api_summary(day):
    """Return the daily_summary row for YYYY-MM-DD."""
    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):
    """
    Return route points for a given day as a list of [lat, lon]
    suitable for drawing a Leaflet polyline.
    """
```

```
    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 jsonify([[r["lat"], r["lon"]] for r in rows])


@app.route("/api/stops/<day>")
def api_stops(day):
    """
    Return stop events that START on a given 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 Mode (writes only to shifts table)
# ============================================================

@app.route("/api/shift/active")
def api_shift_active():
    """
    Returns the currently active shift (where end_ts is NULL),
    or {"active": false} if none exists.
    """
    conn = db()
    cur = conn.cursor()
    cur.execute("""
        SELECT *
        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(dict(row))


@app.route("/api/shift/start", methods=["POST"])
def api_shift_start():
    """
    Start a new shift.
    Prevents multiple active shifts at once.
```

```
    Optional JSON body:
      {"note": "optional note here"}
    """
    note = ""
    try:
        payload = request.get_json(silent=True) or {}
        note = payload.get("note", "") or ""
    except Exception:
        note = ""

    conn = db()
    cur = conn.cursor()

    # Block starting a shift if one is already active
    cur.execute("SELECT id FROM shifts WHERE end_ts IS NULL LIMIT 1")
    if cur.fetchone():
        conn.close()
        return jsonify({"error": "A shift is already active."}), 409

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

    cur.execute("SELECT * FROM shifts WHERE id = last_insert_rowid()")
    row = cur.fetchone()
    conn.close()

    return jsonify(dict(row))


@app.route("/api/shift/end", methods=["POST"])
def api_shift_end():
    """
    End the currently active shift by setting end_ts.
    """
    conn = db()
    cur = conn.cursor()
```

```
    cur.execute("""
        SELECT *
        FROM shifts
        WHERE end_ts IS NULL
        ORDER BY id DESC
        LIMIT 1
    """)
    row = cur.fetchone()

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

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

    cur.execute("SELECT * FROM shifts WHERE id = ?", (row["id"],))
    updated = cur.fetchone()
    conn.close()

    return jsonify(dict(updated))


@app.route("/api/shift/summary")
def api_shift_summary():
    """
    Returns a lightweight summary for the ACTIVE shift window.

    Current output:
      - shift window (start -> now)
      - number of gps points in that window (mode=3)
      - stop count + stopped seconds for stop_events inside window

    NOTE:
      This does not compute miles yet. That comes next.
    """
    conn = db()
    cur = conn.cursor()
```

```
# Find active shift
cur.execute("""
    SELECT *
    FROM shifts
    WHERE end_ts IS NULL
    ORDER BY id DESC
    LIMIT 1
""")
shift = cur.fetchone()

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

start_ts = shift["start_ts"]
end_ts = utc_now_iso()  # "now" for active shift

# gps_points stores timestamps with trailing "Z"
# shifts stores timestamps with "+00:00"
# Convert bounds for gps_points query
start_bound = start_ts.replace("+00:00", "Z")
end_bound = end_ts.replace("+00:00", "Z")

cur.execute("""
    SELECT COUNT(*) as point_count
    FROM gps_points
    WHERE ts >= ? AND ts <= ?
      AND mode = 3
      AND lat IS NOT NULL
      AND lon IS NOT NULL
""", (start_bound, end_bound))
point_row = cur.fetchone()

cur.execute("""
    SELECT COUNT(*) as stop_count,
           COALESCE(SUM(duration_seconds), 0) as stopped_s
    FROM stop_events
    WHERE start_ts >= ? AND end_ts <= ?
""", (start_ts, end_ts))
```

```
    stop_row = cur.fetchone()

    conn.close()

    return jsonify({
        "shift_id": shift["id"],
        "start_ts": start_ts,
        "end_ts": end_ts,
        "points": int(point_row["point_count"] or 0),
        "stop_count": int(stop_row["stop_count"] or 0),
        "stopped_time_seconds": int(stop_row["stopped_s"] or 0),
        "note": shift["note"] or ""
    })



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

Make executable:

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

Restart dashboard service:

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

Dashboard UI Enhancements (Full `index.html`)

The dashboard top bar now includes Shift controls:

- Start Shift
- End Shift
- Refresh Shift
- Shift status pill

A new **Active Shift** section shows live stats and refreshes every 30 seconds.

Replace `/opt/routetrack/web/templates/index.html`

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>
    body { margin: 0; font-family: Arial, sans-serif; }
    #topbar { padding: 10px; background: #111; color: #fff; display: flex; gap: 10px; align-items: center; flex-wrap: wrap; }
    #topbar button { cursor: pointer; }
    #map { height: 70vh; }
    #stats { padding: 10px; }
    .row { margin: 6px 0; }
    code { background: #eee; padding: 2px 4px; border-radius: 4px; }
    .pill { display: inline-block; padding: 2px 8px; border-radius: 999px; font-size: 12px; background: #333; color: #fff; }
  </style>
</head>
```

```
<body>
  <div id="topbar">
    <strong>RouteTrack</strong> — Local Dashboard

    <span>|</span>

    <span>Date:</span>
    <input id="day" type="date" />
    <button onclick="loadAll()">Load Day</button>

    <span>|</span>

    <button onclick="startShift()">Start Shift</button>
    <button onclick="endShift()">End Shift</button>
    <button onclick="loadShift()">Refresh Shift</button>

    <span id="shiftStatus" class="pill">Shift: Unknown</span>
  </div>

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

  <div id="stats">
    <h3>Active Shift</h3>
    <div id="shift"></div>

    <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 shiftDiv = document.getElementById("shift");
  const shiftStatus = document.getElementById("shiftStatus");
```

```
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 = [];

// ----------------------------
// Day-Based Views
// ----------------------------
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) {
  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>
    `;
  }

  // ----------------------------
  // Shift Mode
  // ----------------------------
  async function startShift() {
    const res = await fetch("/api/shift/start", {
      method: "POST",
      headers: {"Content-Type": "application/json"},
      body: JSON.stringify({ note: "" })
    });

    const data = await res.json();
    if (!res.ok) {
      alert(data.error || "Failed to start shift");
      return;
    }
    await loadShift();
  }

  async function endShift() {
    const res = await fetch("/api/shift/end", { method: "POST" });
    const data = await res.json();
    if (!res.ok) {
      alert(data.error || "Failed to end shift");
      return;
    }
    await loadShift();
  }

  async function loadShift() {
    shiftDiv.innerHTML = "";

    const activeRes = await fetch("/api/shift/active");
    const active = await activeRes.json();

    if (!active || active.active === false || !active.id) {
```

```
      shiftStatus.textContent = "Shift: Inactive";
      shiftDiv.innerHTML = "<div class='row'>No active shift. Click <strong>Start Shift</strong> to begin.</div>
";
      return;
    }

    shiftStatus.textContent = `Shift: ACTIVE (#${active.id})`;

    const sumRes = await fetch("/api/shift/summary");
    const s = await sumRes.json();

    if (s.error) {
      shiftDiv.innerHTML = `<div class='row'>${s.error}</div>`;
      return;
    }

    shiftDiv.innerHTML = `
      <div class="row"><strong>Shift ID:</strong> ${s.shift_id}</div>
      <div class="row"><strong>Start (UTC):</strong> <code>${s.start_ts}</code></div>
      <div class="row"><strong>Now (UTC):</strong> <code>${s.end_ts}</code></div>
      <div class="row"><strong>Stops (inside window):</strong> ${s.stop_count}</div>
      <div class="row"><strong>Stopped Minutes:</strong> ${Math.round(s.stopped_time_seconds / 60)}</div
>
      <div class="row"><strong>GPS Points (mode=3):</strong> ${s.points}</div>
    `;
  }

  // Auto-load on page open
  loadAll();
  loadShift();

  // Auto-refresh active shift every 30s (handy for truck use)
  setInterval(loadShift, 30000);
</script>
</body>
</html>
```

Restart dashboard:

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

Validation & Testing

Before starting a shift:

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

Expected:

```
{"active":false}
```

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

Expected:

```
{"error":"No active shift."}
```

Dashboard validation:

- Start Shift creates an active session
- Refresh Shift updates live metrics
- End Shift closes session cleanly
- Status pill returns to “Shift: Inactive”

Result

RouteTrack now supports:

- Continuous GPS logging
- Stop detection + daily summaries
- Local dashboard (Flask + Leaflet)
- **Shift Mode with user-controlled Start/End**
- Live shift summary panel in the UI

This moves RouteTrack closer to a true **truck-ready route tracking + session logging system**.

Next Steps

Now that Shift Mode works end-to-end, next upgrades will add:

1. **Shift mileage + moving time**
   - Apply Haversine logic inside shift window
2. **Persist final shift totals**
   - Save a shift summary row when ending a shift
3. **Shift history**
   - List past shifts and export (CSV/GeoJSON)
4. Optional: offline map tiles

Revision #2
Created 25 December 2025 16:41:13 by Nate Nash
Updated 25 December 2025 21:45:54 by Nate Nash