13 - RouteTrack Pi — Mobile Dashboard Overhaul + BIG LIVE MODE (Stop Point)

Date: December 25, 2025
Category: UI/UX / Mobile Optimization / Dashboard

Backlink: Update #12 – RouteTrack Pi — Mobile UX Upgrade (Shift Controls + Status)

Goal of This Update

This update was focused on making the RouteTrack dashboard **actually usable on mobile** (Firefox on Pixel 10 Pro XL), without sacrificing any of the power/features that were added during earlier iterations.

The main objective was:

- **Mobile-first layout**
- **Shift controls always easy to tap**
- A single, one-press **BIG LIVE MODE** for “driving glance” usage
- Keep the map **contained inside its UI box**
- Preserve all existing RouteTrack dashboard functionality (shift summary, daily summary, stops, filtering, speed-colored route, live polling, etc.)

This is a **good stopping point** for the project until real-world usage exposes what should be improved next.

What Was Added / Improved

Mobile UX + Layout Improvements

- Rebuilt the layout to be **mobile-first** (not desktop-first).
- The top UI was redesigned into a **compact stacked header** with:
  - Date picker + Load
  - Shift controls block
  - Collapsible controls drawer

BIG LIVE MODE (One-Press “Driving View”)

- Added a dedicated **BIG LIVE MODE** button that:
  - Hides all nonessential UI
  - Expands the map to **full screen**
  - Keeps **LIVE MPH** visible at all times
  - Ensures Live Mode is ON (auto-enables if needed)
  - Provides an always-available **Exit** button

Live MPH + Live Status HUD

- A floating HUD on the map displays:
  - **LIVE MPH**
  - Live status (ON / OFF / NO FIX / ERR)
- The HUD is set to `pointer-events: none` so it **won’t block map gestures** on mobile.

Controls Drawer (Mobile Only)

- A **Controls ▾** drawer was added to reduce clutter on mobile.
- This keeps the map usable without constantly fighting UI height.

Preserved Features (No Regressions)

This update explicitly **kept all previously implemented functionality**:

- Speed-colored route rendering (blue/amber/red)
- Daily Summary (with filtered stop totals + raw DB totals)
- Stops list and map markers

- Hide individual stops (saved in browser localStorage)
- Filter stops above X minutes + Clear Hidden Stops
- Live mode polling from `/api/live/latest`
- Shift start/end controls
- Shift status + shift summary refresh loop

Files Modified

Dashboard HTML Template

- **File:** `/opt/routetrack/web/templates/index.html`
- **Purpose:** Main mobile-first UI + map + controls + live mode + summaries

---

Service Restart (Apply Changes)

After updating the template:

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

Optional: If the browser still shows the old UI on mobile, fully close the tab and reopen (mobile Firefox can be aggressive with caching).

Current Project Status

This update marks a clean **pause point** for the RouteTrack Pi project.

The dashboard is now:

- Useable on mobile
- Stable enough to run day-to-day
- Ready for real-world testing (actual driving + shift usage)

Future improvements will be based on real usage (accuracy, stop detection, distance consistency, etc.).

Full Script: `index.html` (Final Version for Update #13)

Location:

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

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

  <!-- Leaflet -->
  <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:#0b0b0b;
     --panel:#121212;
     --panel2:#0f0f0f;
     --border:#222;
     --muted: rgba(255,255,255,0.78);
     --text:#eaeaea;
     --btn:#1a1a1a;
     --btnHover:#222;
     --ok:#86efac;
     --warn:#fca5a5;
     --blue:#3b82f6;
     --amber:#f59e0b;
     --red:#ef4444;
     --chip:#171717;
```

```
  }

  * { box-sizing: border-box; }
  body { margin:0; font-family: system-ui, -apple-system, Segoe UI, Roboto, Arial, sans-serif; background:var(--bg); color:var(--text); }

  .muted { color: var(--muted); }
  .ok { color: var(--ok); }
  .warn { color: var(--warn); }
  .tiny { font-size: 0.9em; }
  .brand { font-weight: 900; letter-spacing: 0.4px; user-select:none; }
  .hr { height:1px; background: var(--border); margin: 10px 0; }

  button {
    cursor:pointer;
    border:1px solid var(--border);
    background: var(--btn);
    color:#fff;
    padding: 10px 12px;
    border-radius: 12px;
    font-weight: 900;
    line-height: 1;
    white-space: nowrap;
  }
  button:hover { background: var(--btnHover); }
  button:disabled { opacity:0.5; cursor:not-allowed; }

  input[type="date"],
  input[type="number"]{
    background: var(--btn);
    color:#fff;
    border:1px solid var(--border);
    border-radius: 12px;
    padding: 9px 10px;
    font-weight: 900;
    max-width: 100%;
  }

  .toggle {
    display:inline-flex; gap:8px; align-items:center;
```

```
  padding: 9px 10px;
  border: 1px solid var(--border);
  border-radius: 999px;
  background: var(--btn);
  font-weight: 900;
}
.toggle input { transform: scale(1.15); }

.chip{
  display:inline-flex;
  align-items:center;
  gap: 10px;
  padding: 10px 10px;
  border-radius: 14px;
  background: var(--chip);
  border: 1px solid var(--border);
  flex-wrap: wrap;
  max-width: 100%;
}

.legendRow{
  display:flex;
  gap: 12px;
  flex-wrap: wrap;
  align-items:center;
  font-size: 0.95em;
}
.swatch { font-weight: 900; width: 14px; display:inline-block; text-align:center; }

/* ====== APP SHELL ====== */
#shell{
  display:flex;
  flex-direction: column;
  min-height: 100vh;
  background: var(--bg);
}

/* ====== TOPBAR (Mobile-first, compact) ====== */
#topbar{
  background: var(--panel);
```

```
  border-bottom: 1px solid var(--border);
  padding: 10px 12px;
  display:flex;
  flex-direction: column;
  gap: 10px;
  position: static; /* mobile: NOT sticky */
  z-index: 1000;
}

.topRow{
  display:flex;
  align-items:center;
  gap: 10px;
  flex-wrap: wrap;
}

.grow { flex: 1 1 auto; }

/* Shift badge (mobile template vibe) */
.shiftBadge{
  padding: 10px 12px;
  border-radius: 14px;
  border: 1px solid var(--border);
  background: #101010;
  font-weight: 900;
  display:flex;
  gap: 10px;
  align-items:center;
  flex-wrap: wrap;
  justify-content: space-between;
  width: 100%;
}

.shiftBadge .left{
  display:flex;
  gap: 10px;
  align-items:center;
  flex-wrap: wrap;
}
```

```
.bigBtn{
  padding: 14px 14px;
  border-radius: 16px;
  font-weight: 950;
  font-size: 1rem;
}

.btnDanger { border-color: #3b1b1b; background: #1a1010; }
.btnDanger:hover { background: #241010; }
.btnAccent { border-color:#12305f; background:#0f1a2a; }
.btnAccent:hover{ background:#13213a; }

/* Controls drawer (collapsed on mobile, open by choice) */
#controlsDrawer{
  display:none;
  flex-direction: column;
  gap: 10px;
  padding: 10px;
  border: 1px solid var(--border);
  border-radius: 14px;
  background: #101010;
}
#controlsDrawer.open{ display:flex; }

/* ====== MAP ====== */
#mapWrap{
  position: relative;
  background: var(--bg);
  border-bottom: 1px solid var(--border);
  overflow: hidden; /* keep map + controls contained */
}

/* dvh fixes mobile toolbar height weirdness */
#map{ width:100%; height: 62dvh; }

/* Floating MPH + Live status */
#hud{
  position: absolute;
  right: 12px;
  top: 12px;
```

```
  z-index: 650;
  display:flex;
  flex-direction: column;
  gap: 8px;
  align-items: flex-end;
  pointer-events: none; /* do not block map gestures */
}

.hudBox{
  padding: 10px 12px;
  border-radius: 14px;
  border: 1px solid var(--border);
  background: rgba(16,16,16,0.92);
  backdrop-filter: blur(4px);
  font-weight: 950;
  display:flex;
  gap: 10px;
  align-items: baseline;
  min-width: 165px;
  justify-content: center;
}
.hudBox small{ font-weight: 800; font-size: 0.8rem; color: var(--muted); }

/* ====== CONTENT BELOW MAP ====== */
#content{
  padding: 12px;
  display:grid;
  grid-template-columns: 1fr;
  gap: 12px;
}

.card{
  background: var(--panel2);
  border: 1px solid var(--border);
  border-radius: 14px;
  padding: 12px;
  box-shadow: 0 10px 24px rgba(0,0,0,0.25);
}

h3{ margin:0 0 10px; font-size: 1.05rem; }
```

```
.grid2 { display:grid; grid-template-columns: 1fr 1fr; gap: 10px; }
@media (max-width: 420px) { .grid2 { grid-template-columns: 1fr; } }

code {
  background: #1f1f1f;
  padding: 2px 6px;
  border-radius: 6px;
  color: #d7d7d7;
  word-break: break-word;
}

.stopItem {
  padding: 10px;
  border-radius: 12px;
  border: 1px solid var(--border);
  background: #101010;
  display:flex;
  flex-direction: column;
  gap: 8px;
}
.stopActions { display:flex; gap: 8px; flex-wrap: wrap; }
.btnSmall { padding: 10px 10px; border-radius: 12px; font-weight: 950; }

/* ====== BIG LIVE MODE ====== */
body.bigLive #topbar,
body.bigLive #content{
  display:none;
}
body.bigLive #map{
  height: 100dvh;
}
body.bigLive #mapWrap{
  border-bottom: none;
}
#exitBigLive{
  position:absolute;
  left: 12px;
  top: 12px;
  z-index: 700;
```

```
      pointer-events: auto;
      display:none;
    }
    body.bigLive #exitBigLive{ display:inline-flex; }

    /* ====== DESKTOP ENHANCEMENTS ====== */
    @media (min-width: 900px){
      #topbar{
        position: sticky;
        top: 0;
      }
      #map{ height: 68vh; }
      #content{
        grid-template-columns: 1fr 1fr;
        align-items:start;
      }
      #controlsDrawer{
        display:flex;
        flex-direction: row;
        align-items:center;
        flex-wrap: wrap;
        gap: 10px;
        padding: 0;
        border: none;
        background: transparent;
      }
      #btnControlsToggle{ display:none; }
      .shiftBadge{ width:auto; }
    }
  </style>
</head>

<body>
<div id="shell">

  <div id="topbar">
    <div class="topRow">
      <span class="brand">RouteTrack</span>
      <span class="muted tiny">Local Dashboard</span>
```

```
      <span class="chip">
        <span class="muted">Date</span>
        <input id="day" type="date" />
        <button onclick="loadAll()">Load</button>
      </span>

      <button id="btnControlsToggle" class="btnAccent" onclick="toggleControls()">Controls ▾</button>
    </div>

    <div class="shiftBadge">
      <div class="left">
        <span id="shiftStatus" class="muted">Shift: …</span>
        <span id="shiftSince" class="muted tiny"></span>
      </div>

      <div class="topRow">
        <button id="btnStartShift" class="bigBtn" onclick="startShift()">Start Shift</button>
        <button id="btnEndShift" class="bigBtn btnDanger" onclick="endShift()">End Shift</button>
        <button id="btnBigLive" class="bigBtn btnAccent" onclick="toggleBigLiveMode()">BIG LIVE
MODE</button>
      </div>
    </div>

    <div id="controlsDrawer">
      <div class="chip legendRow">
        <span><span class="swatch" style="color:var(--blue);">■</span> &lt; 5 mph</span>
        <span><span class="swatch" style="color:var(--amber);">■</span> 5–25 mph</span>
        <span><span class="swatch" style="color:var(--red);">■</span> &gt; 25 mph</span>
      </div>

      <div class="chip">
        <span class="toggle">
          <input id="liveToggle" type="checkbox" onchange="toggleLive()" />
          <label for="liveToggle">Live</label>
        </span>

        <span class="toggle muted">
          <input id="liveCenter" type="checkbox" />
          <label for="liveCenter">Center</label>
        </span>
```

```
      <span id="liveStatus" class="muted">Live: off</span>
    </div>

    <div class="chip grow">
      <span class="muted">Stop Filter</span>

      <span class="toggle muted">
        <input id="ignoreLongStopsToggle" type="checkbox" checked
onchange="renderStopsAndFilteredSummary()" />
        <label for="ignoreLongStopsToggle">Ignore &gt;</label>
      </span>

      <input id="ignoreLongStopsMinutes" type="number" min="0" step="10" value="240"
            title="Stops longer than this many minutes will be ignored for display + summary."
            onchange="renderStopsAndFilteredSummary()" />

      <span class="muted tiny">min</span>

      <button class="btnSmall btnDanger" onclick="clearHiddenStops()">Clear Hidden Stops</button>
    </div>
  </div>
 </div>

 <div id="mapWrap">
  <div id="map"></div>
  <button id="exitBigLive" class="btnDanger" onclick="toggleBigLiveMode()">Exit</button>

  <div id="hud">
    <div class="hudBox">
      <small>LIVE MPH</small>
      <span id="mphValue">—</span>
    </div>
    <div class="hudBox">
      <small>LIVE</small>
      <span id="hudLive">OFF</span>
    </div>
  </div>
 </div>
```

```
  <div id="content">

    <div class="card">
      <h3>Shift Summary</h3>
      <div id="shiftSummary" class="muted">Loading shift summary…</div>
      <div class="hr"></div>
      <div class="muted tiny">
        Shift Summary is computed from GPS points between <strong>Start Shift</strong> and <strong>End Shift</strong>.
        If a shift is active, this updates live.
      </div>
    </div>

    <div class="card">
      <h3>Daily Summary</h3>
      <div id="summary"></div>
      <div class="hr"></div>
      <div class="muted tiny">
        Tip: Daily Summary is based on your daily processor run. Shift Summary is the “real-world” view for work sessions.
      </div>
    </div>

    <div class="card">
      <h3>Stops</h3>
      <div class="muted tiny">
        You can hide individual stops (saved in your browser on this device). Hidden/filtered stops won’t count in the UI totals.
      </div>
      <div class="hr"></div>
      <div id="stops"></div>
    </div>

    <div class="card">
      <h3>Quick Checks</h3>
      <div class="tiny muted">
        • If distance looks wrong for the date, re-run the processor for that date.<br>
        • For “real” totals (crossing midnight), rely on Shift Summary.<br>
        • Live MPH is from latest logged point in <code>gps_points</code>.
      </div>
```

```
    </div>

  </div>
</div>

<script>
  function toggleControls(){
    const d = document.getElementById("controlsDrawer");
    d.classList.toggle("open");
    const btn = document.getElementById("btnControlsToggle");
    btn.textContent = d.classList.contains("open") ? "Controls ▴" : "Controls ▾";
    setTimeout(() => { map.invalidateSize(true); }, 150);
  }
  function closeControlsOnMobile(){
    if (window.matchMedia("(max-width: 899px)").matches){
      const d = document.getElementById("controlsDrawer");
      const btn = document.getElementById("btnControlsToggle");
      d.classList.remove("open");
      btn.textContent = "Controls ▾";
    }
  }

  function toggleBigLiveMode(){
    document.body.classList.toggle("bigLive");
    if (document.body.classList.contains("bigLive")){
      if (!liveToggle.checked){
        liveToggle.checked = true;
        toggleLive();
      }
    }
    setTimeout(() => map.invalidateSize(true), 200);
  }

  const dayInput = document.getElementById("day");
  dayInput.valueAsDate = new Date();

  const ignoreLongStopsToggle = document.getElementById("ignoreLongStopsToggle");
  const ignoreLongStopsMinutes = document.getElementById("ignoreLongStopsMinutes");

  const btnStartShift = document.getElementById("btnStartShift");
```

```
const btnEndShift   = document.getElementById("btnEndShift");
const shiftStatus   = document.getElementById("shiftStatus");
const shiftSince    = document.getElementById("shiftSince");
const shiftSummaryDiv = document.getElementById("shiftSummary");

const liveToggle = document.getElementById("liveToggle");
const liveCenter = document.getElementById("liveCenter");
const liveStatus = document.getElementById("liveStatus");

const mphValue = document.getElementById("mphValue");
const hudLive = document.getElementById("hudLive");

const map = L.map("map", { zoomControl: false }).setView([38.7153, -89.94], 13);
L.control.zoom({ position: "bottomright" }).addTo(map);
L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
  maxZoom: 19,
  attribution: "&copy; OpenStreetMap contributors"
}).addTo(map);

const MPS_TO_MPH = 2.23694;

let routeBoundsHelper = null;
let routeSegments = [];
let stopMarkers = [];

let liveMarker = null;
let liveTimer = null;
let shiftTimer = null;

let stopsCache = [];
let summaryCache = null;

const LS_HIDDEN_STOPS_KEY = "routetrack_hidden_stops_v1";

function getHiddenStopsSet() {
  try {
    const raw = localStorage.getItem(LS_HIDDEN_STOPS_KEY);
    if (!raw) return new Set();
    return new Set(JSON.parse(raw));
  } catch { return new Set(); }
```

```
}

function saveHiddenStopsSet(setObj) {
  localStorage.setItem(LS_HIDDEN_STOPS_KEY, JSON.stringify(Array.from(setObj)));
}

function stopKey(s) {
  return `${s.start_ts}|${s.end_ts}|${s.lat}|${s.lon}|${s.duration_seconds}`;
}

function clearHiddenStops() {
  localStorage.removeItem(LS_HIDDEN_STOPS_KEY);
  renderStopsAndFilteredSummary();
}

function fmtMph(speedMps) {
  if (speedMps === null || speedMps === undefined) return "—";
  const mph = speedMps * MPS_TO_MPH;
  return mph.toFixed(1);
}

function fmtLocal(ts) {
  try {
    const d = new Date(ts);
    if (isNaN(d.getTime())) return ts;
    return d.toLocaleString(undefined, {
      year: "numeric", month: "2-digit", day: "2-digit",
      hour: "2-digit", minute: "2-digit"
    });
  } catch { return ts; }
}

function speedStyle(speedMps) {
  if (speedMps === null || speedMps === undefined) return { color: "#777", weight: 5, opacity: 0.7 };
  const mph = speedMps * MPS_TO_MPH;
  if (mph < 5)  return { color: "var(--blue)",  weight: 5, opacity: 0.85 };
  if (mph < 25) return { color: "var(--amber)", weight: 5, opacity: 0.9 };
  return          { color: "var(--red)",   weight: 5, opacity: 0.9 };
}
```

```
function getPanPadding() {
  const isDesktop = window.matchMedia("(min-width: 900px)").matches;
  const topbar = document.getElementById("topbar");
  const top = (isDesktop && topbar) ? topbar.offsetHeight : 0;
  return {
    paddingTopLeft: [20, top + 20],
    paddingBottomRight: [20, 40]
  };
}

async function loadAll() {
  const day = dayInput.value;
  await Promise.all([
    loadRoute(day),
    loadSummary(day),
    loadStops(day),
    refreshShiftStatus(),
    refreshShiftSummary(),
  ]);
  closeControlsOnMobile();
}

async function loadRoute(day) {
  routeSegments.forEach(seg => map.removeLayer(seg));
  routeSegments = [];
  if (routeBoundsHelper) { map.removeLayer(routeBoundsHelper); routeBoundsHelper = null; }

  const res = await fetch(`/api/points_detailed/${day}`);
  const pts = await res.json();
  if (!Array.isArray(pts) || pts.length < 2) return;

  for (let i = 1; i < pts.length; i++) {
    const a = pts[i - 1];
    const b = pts[i];
    const seg = L.polyline([[a.lat, a.lon], [b.lat, b.lon]], speedStyle(b.speed)).addTo(map);
    routeSegments.push(seg);
  }

  const latlngs = pts.map(p => [p.lat, p.lon]);
  routeBoundsHelper = L.polyline(latlngs, { opacity: 0 }).addTo(map);
```

```
    map.fitBounds(routeBoundsHelper.getBounds(), { padding: [18, 18] });
  }

  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) {
      summaryCache = null;
      summaryDiv.innerHTML = `<div class="muted">No summary for ${day}. Re-run processor for that
date.</div>`;
      return;
    }

    summaryCache = data;
    renderStopsAndFilteredSummary();
  }

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

    const res = await fetch(`/api/stops/${day}`);
    const stops = await res.json();
    stopsCache = Array.isArray(stops) ? stops : [];
    renderStopsAndFilteredSummary();
  }

  function getFilteredStops() {
    const hidden = getHiddenStopsSet();
    const ignoreLong = ignoreLongStopsToggle.checked;
    const cutoffMin = Number(ignoreLongStopsMinutes.value || 0);

    return stopsCache.filter(s => {
      const k = stopKey(s);
      if (hidden.has(k)) return false;
      if (ignoreLong && cutoffMin > 0) {
```

```
      const durMin = Number(s.duration_seconds) / 60;
      if (durMin > cutoffMin) return false;
    }
    return true;
  });
}

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

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

  const filtered = getFilteredStops();

  if (!stopsCache.length) {
    stopsDiv.innerHTML = "<div class='muted'>No stops found.</div>";
  } else if (!filtered.length) {
    stopsDiv.innerHTML = "<div class='muted'>All stops are hidden/filtered.</div>";
  } else {
    filtered.forEach(s => {
      const durMin = Math.round(Number(s.duration_seconds) / 60);

      const item = document.createElement("div");
      item.className = "stopItem";
      item.innerHTML = `
        <div>
          <div><strong>Stop</strong> • ${durMin} min</div>
          <div class="tiny muted">Start: <code>${fmtLocal(s.start_ts)}</code></div>
          <div class="tiny muted">End: <code>${fmtLocal(s.end_ts)}</code></div>
          <div class="tiny muted">Coords: <code>${Number(s.lat).toFixed(6)},
${Number(s.lon).toFixed(6)}</code></div>
        </div>
        <div class="stopActions">
          <button class="btnSmall" data-action="zoom">Zoom</button>
          <button class="btnSmall btnDanger" data-action="hide">Hide</button>
        </div>
      `;
```

```
      item.querySelector('button[data-action="zoom"]').onclick = () => {
        map.setView([s.lat, s.lon], Math.max(map.getZoom(), 16), { animate: true });
      };

      item.querySelector('button[data-action="hide"]').onclick = () => {
        const setObj = getHiddenStopsSet();
        setObj.add(stopKey(s));
        saveHiddenStopsSet(setObj);
        renderStopsAndFilteredSummary();
      };

      stopsDiv.appendChild(item);

      const popup = `
        <div style="min-width:230px">
          <div><strong>Stop</strong> (${durMin} min)</div>
          <div style="margin-
top:6px"><strong>Start:</strong><br><code>${fmtLocal(s.start_ts)}</code></div>
          <div style="margin-
top:6px"><strong>End:</strong><br><code>${fmtLocal(s.end_ts)}</code></div>
          <div style="margin-top:6px"><strong>Coords:</strong> ${Number(s.lat).toFixed(6)},
${Number(s.lon).toFixed(6)}</div>
        </div>
      `;
      const m = L.marker([s.lat, s.lon]).addTo(map).bindPopup(popup);
      stopMarkers.push(m);
    });
  }

  renderDailySummaryWithFilteredStops(filtered);
}

function renderDailySummaryWithFilteredStops(filteredStops) {
  const summaryDiv = document.getElementById("summary");
  summaryDiv.innerHTML = "";

  if (!summaryCache) {
    summaryDiv.innerHTML = `<div class="muted">No daily summary loaded.</div>`;
    return;
  }
```

```
    const filteredStopCount = filteredStops.length;
    const filteredStoppedSeconds = filteredStops.reduce((acc, s) => acc + Number(s.duration_seconds || 0), 0);

    const movingMin = Math.round(Number(summaryCache.moving_time_seconds || 0) / 60);
    const stoppedMin = Math.round(filteredStoppedSeconds / 60);

    summaryDiv.innerHTML = `
      <div class="grid2">
        <div><span
class="muted">Start</span><br><code>${fmtLocal(summaryCache.start_ts)}</code></div>
        <div><span
class="muted">End</span><br><code>${fmtLocal(summaryCache.end_ts)}</code></div>
      </div>

      <div class="hr"></div>

      <div class="grid2">
        <div><span
class="muted">Distance</span><br><strong>${summaryCache.total_distance_miles}</strong> miles</div>
        <div><span class="muted">Moving</span><br><strong>${movingMin}</strong> minutes</div>
      </div>

      <div class="grid2">
        <div><span class="muted">Stopped (filtered)</span><br><strong>${stoppedMin}</strong>
minutes</div>
        <div><span class="muted">Stops (filtered)</span><br><strong>${filteredStopCount}</strong></div>
      </div>

      <div class="tiny muted">
        Raw DB: stopped <code>${Math.round(Number(summaryCache.stopped_time_seconds||0)/60)}</code>
min •
        stops <code>${summaryCache.stop_count}</code>
      </div>
    `;
  }

  async function refreshShiftStatus() {
    try {
      const res = await fetch("/api/shift/status");
```

```
    const data = await res.json();

    if (data.active) {
      btnStartShift.disabled = true;
      btnEndShift.disabled = false;
      shiftStatus.innerHTML = `Shift: <span class="ok">ACTIVE</span>`;
      shiftSince.innerHTML = `since <code>${fmtLocal(data.shift.start_ts)}</code>`;
    } else {
      btnStartShift.disabled = false;
      btnEndShift.disabled = true;
      shiftStatus.innerHTML = `Shift: <span class="warn">INACTIVE</span>`;
      shiftSince.textContent = "";
    }
  } catch {
    shiftStatus.textContent = "Shift: error";
    shiftSince.textContent = "";
  }
}

async function startShift() {
  btnStartShift.disabled = true;
  try {
    const res = await fetch("/api/shift/start", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({})
    });
    const text = await res.text();
    let data = {};
    try { data = JSON.parse(text); } catch {}
    if (!res.ok) alert((data && data.error) ? data.error : `Failed (HTTP ${res.status})`);
  } catch {
    alert("Failed to start shift (request failed)");
  }
  await refreshShiftStatus();
  await refreshShiftSummary();
}

async function endShift() {
  btnEndShift.disabled = true;
```

```
    try {
      const res = await fetch("/api/shift/end", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({})
      });
      const text = await res.text();
      let data = {};
      try { data = JSON.parse(text); } catch {}
      if (!res.ok) alert((data && data.error) ? data.error : `Failed (HTTP ${res.status})`);
    } catch {
      alert("Failed to end shift (request failed)");
    }
    await refreshShiftStatus();
    await refreshShiftSummary();
  }

  async function refreshShiftSummary() {
    try {
      const res = await fetch("/api/shift/summary");
      const data = await res.json();
      if (!res.ok || data.error) {
        shiftSummaryDiv.innerHTML = `<div class="muted">No shift summary available yet.</div>`;
        return;
      }

      const sum = data.summary;
      const activeLabel = data.active ? `<span class="ok">ACTIVE</span>` : `<span
class="warn">COMPLETED</span>`;

      shiftSummaryDiv.innerHTML = `
        <div>Shift: ${activeLabel}</div>

        <div class="grid2" style="margin-top:8px;">
          <div><span class="muted">Start</span><br><code>${fmtLocal(sum.start_ts)}</code></div>
          <div><span class="muted">End</span><br><code>${fmtLocal(sum.end_ts)}</code></div>
        </div>

        <div class="hr"></div>
```

```
      <div class="grid2">
        <div><span class="muted">Distance</span><br><strong>${sum.distance_miles}</strong>
miles</div>
        <div><span class="muted">Max Speed</span><br><strong>${sum.max_mph}</strong> mph</div>
      </div>

      <div class="grid2">
        <div><span class="muted">Moving</span><br><strong>${sum.moving_minutes}</strong>
minutes</div>
        <div><span class="muted">Stopped (dwell)</span><br><strong>${sum.stopped_minutes}</strong>
minutes</div>
      </div>

      <div><span class="muted">Stops (dwell)</span><br><strong>${sum.stop_count}</strong></div>

      <div class="tiny muted" style="margin-top:6px;">
        Points processed: <code>${sum.points}</code>
      </div>
    `;
  } catch {
    shiftSummaryDiv.innerHTML = `<div class="muted">Shift summary error.</div>`;
  }
}

async function pollLiveOnce() {
  try {
    const res = await fetch("/api/live/latest");
    const data = await res.json();

    if (!res.ok || data.error) {
      liveStatus.innerHTML = `Live: <span class="warn">no fix</span>`;
      mphValue.textContent = "—";
      hudLive.textContent = "NO FIX";
      return;
    }

    const latlng = [data.lat, data.lon];
    const mph = fmtMph(data.speed);

    mphValue.textContent = mph;
```

```
      hudLive.textContent = "ON";

      const label = `${fmtLocal(data.ts)} • ${mph} mph`;

      if (!liveMarker) {
        liveMarker = L.marker(latlng).addTo(map).bindPopup(label);
      } else {
        liveMarker.setLatLng(latlng);
        liveMarker.setPopupContent(label);
      }

      liveStatus.innerHTML = `Live: <span class="ok">ON</span> • <span class="muted">${mph}
mph</span>`;

      if (liveCenter.checked) {
        map.panInside(latlng, getPanPadding());
      }
    } catch {
      liveStatus.innerHTML = `Live: <span class="warn">error</span>`;
      mphValue.textContent = "—";
      hudLive.textContent = "ERR";
    }
  }

  function toggleLive() {
    if (liveToggle.checked) {
      liveStatus.innerHTML = `Live: <span class="ok">starting…</span>`;
      hudLive.textContent = "ON";
      pollLiveOnce();
      liveTimer = setInterval(pollLiveOnce, 2000);
      if (!shiftTimer) shiftTimer = setInterval(refreshShiftSummary, 5000);
    } else {
      if (liveTimer) clearInterval(liveTimer);
      liveTimer = null;
      liveStatus.textContent = "Live: off";
      mphValue.textContent = "—";
      hudLive.textContent = "OFF";
      if (shiftTimer) { clearInterval(shiftTimer); shiftTimer = null; }
    }
  }
```

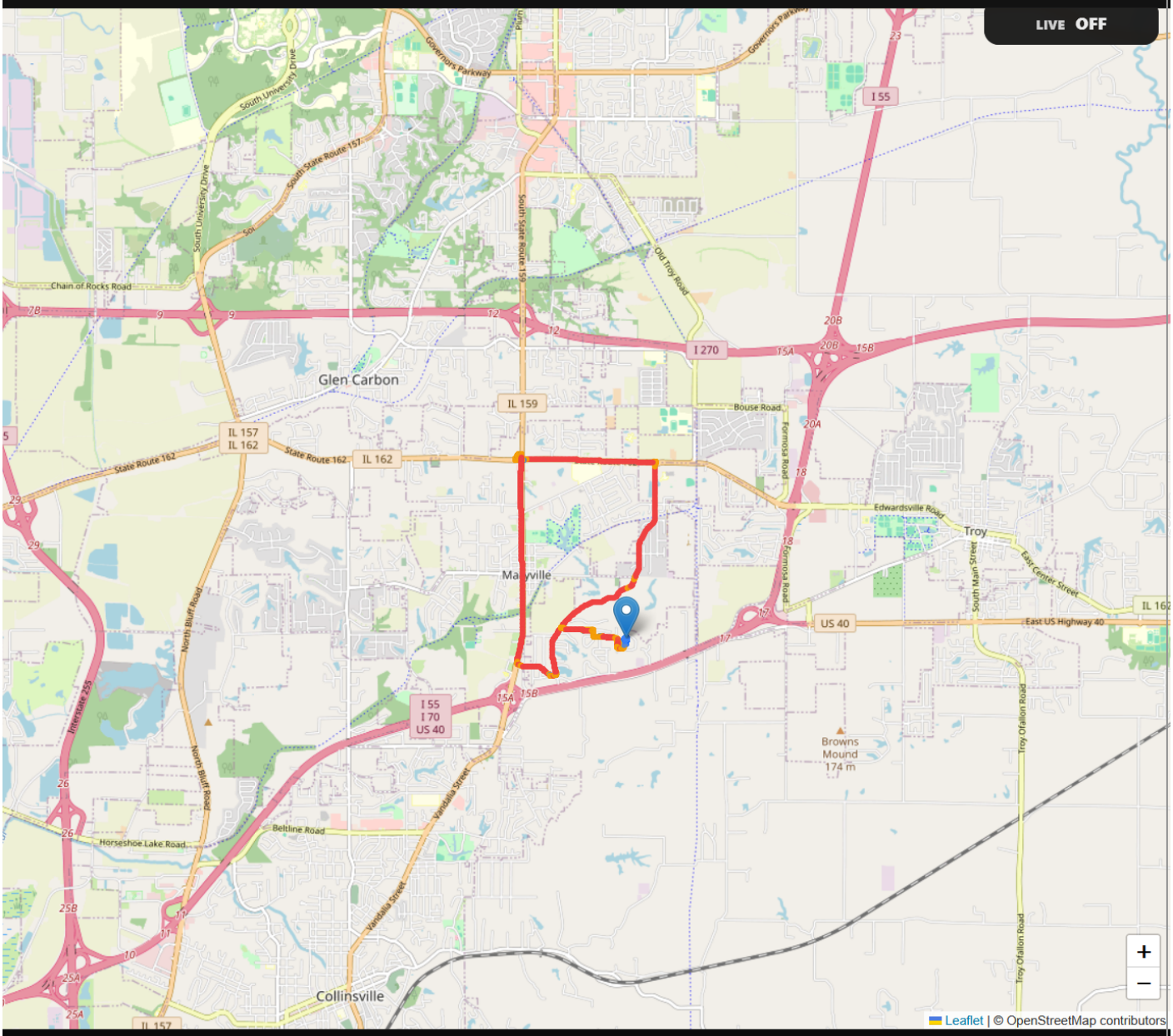
```
  window.addEventListener("resize", () => setTimeout(() => map.invalidateSize(true), 150));

  loadAll();
  setInterval(refreshShiftStatus, 10000);
  setInterval(refreshShiftSummary, 15000);
</script>
</body>
</html>
```

RouteTrack Local Dashboard Date 12 / 25 / 2025 Load

Shift: INACTIVE Start Shift End Shift BIG LIVE MODE

< 5 mph 5–25 mph > 25 mph Live Center Live: off

Stop Filter Ignore > 240 min Clear Hidden Stops



Shift Summary

Shift: COMPLETED

Start
12/25/2025, 03:35 PM

End
12/25/2025, 03:36 PM

Distance
0.001 miles

Max Speed
1.1 mph

Moving
0 minutes

Stopped (dwell)
0 minutes

Stops (dwell)

Daily Summary

Start
12/24/2025, 08:03 PM

End
12/25/2025, 02:36 PM

Distance
7.33 miles

Moving
21 minutes

Stopped (filtered)
0 minutes

Stops (filtered)
0

Raw DB: stopped 1087 min • stops 2

Tip: Daily Summary is based on your daily processor run. Shift Summary is the

Revision #1
Created 25 December 2025 21:42:13 by Nate Nash
Updated 25 December 2025 21:45:38 by Nate Nash