# CLAUDE.md

This file provides guidance to Claude Code (claude.ai/code) when working with code in this repository.

## What this is

A personal indoor-cycling training log (Christopeit AX 4000 + Kinomap), not a software project. No build system, no package manager — just a single ground-truth CSV, a static HTML dashboard that reads it, and a PDF export of that dashboard.

## Architecture: one CSV, everything else derived

`training_log.csv` is the **only** hand-maintained data in this repo. `training.html` is static — it `fetch()`es the CSV at load time, parses it client-side, and derives every chart, table row, hero metric, weekly rollup and legend from it at render time. Adding a session means appending one row to the CSV (via the `add-training` skill) — `training.html` itself never needs to be edited.

- **`training_log.csv`** — one row per session, or per rest-day BP-only log. Columns:
  - `Datum` (`D.M`, no year), `Nr` (session label — plain sequential integer since ~week 9, `-` for rest days; not derivable, always script-assigned)
  - `Dauer_sek`, `Kcal`, `Watt` (estimated, see formula below), `MaxHF`, `Rec` (editorial "strong session" flag, `1` or blank)
  - `RR_ruhe`, `RR_training`, `RR_abend` — blood pressure readings as `SYS/DIA/PULS`, optionally with a trailing `(HH:MM)`/`(Mittag)` annotation. Rest-day rows carry real BP readings too — `Nr:'-'` only means "no training that day", not "no data".
  - `Kinomap`, `Strecke`, `Distanz_km`, `Kadenz_rpm`, `Watt_Max`, `Watt_Max_PR` (□ — set automatically when a session's max watt beats every prior `Watt_Max` in the file), `Hoehenmeter`, `Temperatur_C`, `Max_kmh` — structured fields parsed out of the Kinomap on-screen summary; blank for non-Kinomap sessions.
  - `Kommentar` — free-text remainder (Stufentest/Pyramide labels, warnings, subjective notes) that doesn't fit a structured column above.
  - **Not stored — always computed by `training.html` at render time:** weekday, week number (`Math.floor((date-25.5)/7)+1`, fixed 7-day blocks from 25.5, not ISO weeks), cumulative kcal, week colors (cycled from a fixed palette), week date-range labels, hero metrics, and every chart's axis bounds. There is nothing to keep in sync by hand — if a number ever looks stale, it's a bug in `training.html`'s derivation, not a missed manual edit.
- **`training.html`** — reads `training_log.csv` via `fetch()` and renders all 4 tabs (Täglich/Wöchentlich/Tabelle/Plan 200W) with Chart.js (loaded from `cdnjs.cloudflare.com`, no local deps). Because it does a real `fetch()`, it must be served over HTTP (`python3 -m http.server` from the repo root, then open `http://localhost:<port>/training.html`) — opening the file directly (`file://`) fails silently because Chrome blocks local-file `fetch()` under file:// (CORS). The Plan tab's phase-progress prose cards and the "Ziel" (target watt) series are a hand-authored training plan, not logged data — those stay manually edited and are unaffected by any of this.
- **`training.pdf`** — a print export of `training.html` (all four tabs, light theme, landscape, pure white background), regenerated by hand after editing the CSV — see recipe below. Not needed after a normal `add-training` run, which regenerates it automatically.
- **`stufentest.html` / `stufentest_regression.html`** — standalone step-test (Stufentest) watt/HR calibration charts, unrelated data (`watts[]`/`bpms[]` arrays), used to derive the HR-based watt estimation formula. Untouched by the CSV refactor.
- **`wkg_histogram.html`**, **`training_slideshow.html`** — other standalone single-purpose dashboards with their own embedded data. Untouched.
- **`PulsCharts/`** — a self-contained folder: `puls_graphen_uebersicht.html` plus the watch-screenshot `.jpg` files it displays, referenced by plain relative filename (no subfolder) in the `<img src>`.

## Generating training.pdf

`training.pdf` must be regenerated by hand after editing `training_log.csv` directly (the `add-training` skill does this automatically — this recipe is for a manual/bulk edit). Because `training.html` fetches the CSV, it must be rendered through a local HTTP server, not `file://`.

Recipe (headless Google Chrome + poppler's `pdftoppm`/`pdftotext` for verification):

```
SCRATCH=/path/to/scratch    # any writable temp dir

python3 - <<'EOF'
p = '/Users/haraldbeker/training/training.html'
s = open(p).read()
s = s.replace(
    ".sec{display:none}", ".sec{display:block}"
)   # show all 4 tabs
s = s.replace(
    "const isDark=matchMedia("
```

```
    "'(prefers-color-scheme: dark)').matches;",
    "const isDark=false;",
)
out = '/Users/haraldbeker/training/_training_print_tmp.html'
open(out, 'w').write(s)  # must live in repo root, next to CSV
EOF

PORT=8756   # any free local port, not 5000 (macOS AirPlay)
(cd /Users/haraldbeker/training \
  && python3 -m http.server $PORT >/dev/null 2>&1 &)
sleep 1

CHROME="/Applications/Google Chrome.app/Contents/MacOS/Google Chrome"
URL="http://localhost:$PORT/_training_print_tmp.html"
"$CHROME" \
  --headless --disable-gpu --no-pdf-header-footer \
  --virtual-time-budget=8000 \
  --print-to-pdf="$SCRATCH/final.pdf" "$URL"

cp "$SCRATCH/final.pdf" /Users/haraldbeker/training/training.pdf
rm /Users/haraldbeker/training/_training_print_tmp.html
lsof -ti:$PORT | xargs -r kill
```

Before trusting the output, verify at high DPI — the low-res inline PDF preview is not reliable enough to catch either of the bugs below:

```
pdftoppm -png -r 200 -f 1 -l 1 \
  /Users/haraldbeker/training/training.pdf \
  "$SCRATCH/check"
# crop/zoom the daily "Ø Watt pro Tag" chart and confirm
# the tallest bar's pixel height lines up with its real
# value's gridline.
```

Pitfalls already fixed in training.html — don't reintroduce them:

- **Chart bars/lines can render squashed relative to their own y-axis even when Chart.js's config and data are correct** (e.g. a value of 213 only reaching the "140" gridline on a 100–230 axis). Root cause: Chart.js's default ~1000ms bar-growth animation gets captured mid-animation by the headless PDF snapshot, since `--virtual-time-budget` doesn't reliably advance `requestAnimationFrame`. Fixed by `animation:false` in `baseOpts` (in the script's `render()` function) — every chart spreads `...baseOpts`, so this covers all 9 charts. If squashing ever comes back, check this line first.
- **Chart axis bounds are auto-derived from the actual data on every render** (`scaleMin`/`scaleMax` helpers, ~8% headroom, rounded to a clean step) — this is no longer a hand-maintained value that can silently clip real data past an old max, unlike before this refactor.
- **Print-only CSS overrides** live in `@media print{...}` near the top of `<style>` (pure white `--bg`/`--surface`/`--surface2`, `@page{size:landscape}`, `body{max-width:none}` so the wide session table fits) and later (`.rec-row td{background:none}` — suppresses the "sehr gute Leistung" row highlight in print to save toner; must be declared *after* the base `.rec-row td` rule, since two rules with equal CSS specificity resolve by source order even across separate `@media print` blocks).
- `@media print{.card{break-inside:avoid;page-break-inside:avoid}}` (next to `.card{...}`) stops a chart card from being sliced across a page boundary.
- `_training_print_tmp.html` must be written **inside the repo root** (next to `training_log.csv`), not in `$SCRATCH` — its relative `fetch('training_log.csv')` resolves against whatever URL it's served from.

## Key domain facts

- Watt is estimated from kcal/min, not measured directly: `Watt = (kcal/min × 1000 × 4.185) / 60 × 0.25`, i.e. `Watt = kcal/min × 17.4375`. Inverting it: `Dauer_min = kcal × 17.4375 / Watt`. Displayed with a `~` prefix everywhere, since it's always an estimate.
- Dates are `D.M` with no year (implicit 2026), no leading zeros (e.g. `'5.6'`, `'14.8'`).
- Training weeks are fixed calendar boundaries starting Monday-ish from 25.5, 7-day blocks, not ISO weeks — computed by `getWeek()` in `training.html`'s script, not stored anywhere.
- A day can have multiple sessions (e.g. two rides logged the same date); when aggregating per day, sum kcal and average watt across that day's entries (see `dayMap` in `training.html`'s `render()` function).
- The "Total Dauer" hero/footer figure truncates each session's seconds before summing (not rounding the grand total) — a quirk preserved from the pre-CSV data so the displayed total didn't shift during the migration.