**Sprint 8 – Scaffold & dataadapter (🧩)  
Kontekstblokk  
• Fra S5: Indoor/outdoor + GPS/Wind stabil, CLI-felter (wind\_rel, v\_rel, calibrated, status) på plass.  
• Fra S6: Rapport/observabilitet levert; strukturert JSON-logging og metrikker fungerer.  
• Fra S7: Dynamisk DoD oppdatert (schema\_version, ≥30 samples, edge-case-dekning). Frontend må forvente schema\_version og tåle HR-only fallback.  
• Avhengigheter: Ingen nye tredjepartstjenester; bygger på eksisterende CLI/API (analyze\_session) og testoppsett.  
• Neste steg (Frontlog): S9 – Økt-kort & nøkkelmetrikker.**

**Sprintbeskrivelse  
Oppgaver:**

1. **React/Tailwind scaffold + routing.**
2. **State-management.**
3. **Backend-adapter: bruk analyze\_session() (mock→live).**
4. **🆕 Dokumentasjonstabell for CLI-flagg (navn, type, default, eksempel).  
   DoD (må alle bestås):  
   • Viser én økt (mock) i UI; kan byttes til live via .env.  
   • CLI-flagg-tabell i docs/.  
   • Frontend validerer at schema\_version finnes (semver) og håndterer HR-only uten crash.  
   Estimert tid: 6–9h**

**Arbeidsmetode (for Copilot) – “halvfast default” (justert)**

1. **PLAN (5–7 trinn)  
   • Hvilke filer, hvorfor, forventet output, risikopunkter. Vent på “OK PLAN”.**
2. **Kjør trinnvis  
   • Hvert trinn: diff → kjør tester → vis CLI-output. Vent på “OK <trinn>”.**
3. **Stop-regler (justert for FE-sprint):  
   • Stopp hvis: backend API-semantikk endres (ingen endringer i core/CLI i S8).  
   • Tillatt >10 filer når ≥80 % er FE scaffold/docs. Hvis det foreslås endringer i core/cli → stopp og avklar.  
   • Ingen nye tunge UI-rammeverk; kun Vite/React/Tailwind/Router og nødvendige dev-avhengigheter.**
4. **Schema-bump sjekkliste (ny):  
   • Ingen bump planlagt i S8; frontend skal validere schema\_version (semver).  
   • Hvis bump blir nødvendig: oppdater docs/schema/session\_vX.Y.json (+ kort docs/schema.md).  
   • Sørg for test som validerer tilstedeværelse/semver før merge.**
5. **Debug-partner  
   • Bruk ChatGPT ved uklare FE/ENV-feil eller adapter/validering.**
6. **Ferdigkriterium  
   • Alle DoD verifisert i UI (mock→live) + dokumentasjon (CLI-flagg-tabell).  
   Svarformat (Copilot):  
   • PLAN → vent på “OK PLAN”.  
   • TRINN 1 … → vent på “OK 1”, “OK 2”, osv.**

**Sprint-delta (mini-kontekst)  
• S8 er en FE-scaffold-sprint: UI-skall, enkel state, adapter (mock→live) og dokumentasjon.  
• Ingen endring i fysikkmotor/CLI-logikk; fokus på konsum av eksisterende schema-versionert output.**

**Tekniske rammer (kontrakt)  
• Input: analyze\_session()-JSON (schema\_version, avg\_hr, calibrated, status, ev. wind\_rel, v\_rel, watts).  
• Output: UI som rendrer én økt uten crash, inkl. HR-only fallback.  
• CLI: eksisterende flagg dokumenteres (docs/cli\_flags.md).  
• ENV: VITE\_BACKEND\_URL styrer mock→live.**

**Endringsgrenser  
• Tillatt: frontend/\*\* (Vite/React/Tailwind/Router), docs/cli\_flags.md, .env.example, .gitignore (node\_modules/dist), lett README.  
• Ikke rør: core/src/\*\*, kalibrering/fysikkmotor, Strava-klient, eksisterende backend-tester og CLI-semantikk.**

**Tester / kommandoer  
npm create vite@latest frontend -- --template react-ts  
cd frontend && npm i && npm i -D tailwindcss postcss autoprefixer && npx tailwindcss init -p  
npm i react-router-dom zod && npm run dev**

**Live-switch eksempel:**

**(fra frontend/)**

**echo VITE\_BACKEND\_URL=**[**http://localhost:8000**](http://localhost:8000) **> .env.local && npm run dev**

**(valgfritt) Semver-sjekk av live-endepunkt:**

**curl -s "$VITE\_BACKEND\_URL/api/analyze\_session?id=mock" | python -c "import sys,json,re; d=json.load(sys.stdin); assert re.match(r'^\d+.\d+.\d+$', d.get('schema\_version',''))"**

**Verifikasjonsplan (DoD → test)**

1. **Schema-validering i UI: respons uten schema\_version (semver) gir kontrollert feilmelding (ikke crash).**
2. **Mock-visning: Home/Session-view rendrer én mock-økt med avg\_hr/calibrated/status.**
3. **ENV-switch: .env (VITE\_BACKEND\_URL) bytter til live uten kodeendring.**
4. **HR-only fallback: mock uten watts rendres kontrollert med status/fallback.**
5. **Bygg: npm run build fullfører uten feil.**
6. **Docs: docs/cli\_flags.md finnes med navn, type, default, eksempel.**

**Forventet output (filer/endringer)  
• frontend/\*\* (Vite React TS + Tailwind + Router; src/lib/api.ts & schema.ts; src/routes/*; src/components/SessionCard.tsx; src/state/*; styles).  
• .env.example (frontend) for VITE\_BACKEND\_URL.  
• .gitignore oppdatert for frontend/node\_modules og dist.  
• docs/cli\_flags.md (tabell over CLI-flagg).**

**Frontlog (kommende oppgaver)  
• S9 – Økt-kort & nøkkelmetrikker (📊): UI for NP/IF/VI/Pa:Hr/W/slag/CGS/PrecisionWatt; no-watt varsel; formatter-tester.  
• S10 – Precision Watt UI (🎛️): Graf med usikkerhetsbånd; tooltip (datakilde, vær, profil); performance på 1 Hz/2 t.**

**Gi meg så detaljerte detaljer trinn for trinn at jeg og chat gpt kan gjøre det sammen mens du styrer fremgangen. Det vil si for hvert trinn list opp hvilke filer som er involvert , hva som skal gjøres med de og forventet outptut.**

Lag en Sluttrapport etter Følgende format

✅ Sprint: [ID – Navn]

Branch: [navn]

Commits: [hash – message]

Endrede filer: [paths]

Tester: [pytest + cargo test resultater]

Observasjoner: [korte punkter]

Status: [Ferdig / Delvis ferdig / Feil gjenstår]

**Sluttføring av Workflow (ChatGPT v2)**

1. **Sluttrapport:** lim inn Copilot-rapport i template.
2. **Delta-sammendrag:** skriv 2–3 linjer etter komprimeringsmal:
   * Hva ble gjort.
   * Teststatus.
   * Observasjon.
3. **Evaluering & Oppdatering:** oppdater DoD og Frontlog basert på nye funn/innsikter.
   * Klassifiser funn som A (DoD), B (Frontlog), C (Observasjon).
   * Marker ferdige sprinter, flytt avhengigheter.
   * Juster Sprintplan om nødvendig.
4. **Synk Masterplan:** oppdater statuslinje eller legg til ny rad.
5. **Oppdater Sprint Log:** kort notat med status og carry-over.
6. **Template neste sprint:** sjekk om “halvfast default” trenger justering og fyll ut template basert på neste sprint.