**NOTATER**

**Kort referat (M7 – 4.1 og 4.2)**

* **4.1 – Strava formatter: Lagde cli/formatters/strava\_publish.py som genererer kort kommentar (≤280) og beskrivelses‑header (≤140) på NO/EN, med trimming, fallbacks og uten hengende separator. Testet via tests/test\_strava\_publish.py.**
* **4.2 – CLI‑integrasjon (dry‑run):**
  + **Oppdatert cli/analyze.py med robust kjerne‑shim (støtter Api.analyze\_session[\_json], analyze\_session(...), og legacy calculate\_efficiency\_series(...)).**
  + **Lagt til --dry-run og --lang (NO/EN).**
  + **Dry‑run printer full rapport + [DRY-RUN] COMMENT og [DRY-RUN] DESC.**
  + **Verifisert kjøring mot data/streams/\*.csv.**

**4) Strava‑publisering (kommentar/description)**

**4.1 Opprett formatter‑modul**

**Fil:** cli/formatters/strava\_publish.py

python

KopierRediger

from dataclasses import dataclass

from typing import Optional, Dict

STRAVA\_COMMENT\_MAX = 280

STRAVA\_DESC\_HEADER\_MAX = 140

ARROWS = {"up":"↑", "down":"↓"}

ELLIPSIS = "…"

@dataclass

class PublishPieces:

comment: str

desc\_header: str

desc\_body: Optional[str]

def \_pct(v: Optional[float], decimals=1) -> Optional[str]:

if v is None: return None

return f"{v:.{decimals}f}%"

def \_num(v: Optional[float], decimals=2) -> Optional[str]:

if v is None: return None

return f"{v:.{decimals}f}"

def \_trim(s: str, limit: int) -> str:

if len(s) <= limit: return s

return s[:max(0, limit-1)] + ELLIPSIS

def \_lang\_labels(lang: str):

if lang == "en":

return {

"cgs":"CGS", "how\_hard":"How hard", "how\_long":"How long",

"how\_even":"How even & efficient", "trend":"Trend",

"w\_per\_beat":"W/beat", "pa\_hr":"Pa:Hr", "vi":"VI"

}

# default: norsk

return {

"cgs":"CGS", "how\_hard":"Hvor hardt", "how\_long":"Hvor lenge",

"how\_even":"Hvor jevnt & effektivt", "trend":"Trend",

"w\_per\_beat":"W/slag", "pa\_hr":"Pa:Hr", "vi":"VI"

}

def build\_publish\_texts(report: Dict, lang: str="no") -> PublishPieces:

L = \_lang\_labels(lang)

# Hent felter med fallbacks

s = report

scores = s.get("scores", {})

cgs = scores.get("cgs")

intensity = scores.get("intensity")

duration = scores.get("duration")

quality = scores.get("quality")

vi = s.get("vi")

pa = s.get("pa\_hr\_pct")

wpb = s.get("w\_per\_beat")

wpb\_base = s.get("w\_per\_beat\_baseline")

trend = (s.get("trend") or {}).get("cgs\_delta\_vs\_last3")

# Delta W/beat

wpb\_delta\_pct = None

if wpb is not None and wpb\_base:

wpb\_delta\_pct = (wpb - wpb\_base) / wpb\_base \* 100.0

# Kort kommentar (≤280)

parts = []

if cgs is not None: parts.append(f"CycleGraph {L['cgs']} {int(round(cgs))}")

if s.get("if") is not None: parts.append(f"· IF {\_num(s['if'],2)}")

if vi is not None: parts.append(f"· {L['vi']} {\_num(vi,2)}")

if pa is not None: parts.append(f"· {L['pa\_hr']} {\_num(pa,1)}%")

if wpb is not None:

wpb\_piece = f"· {L['w\_per\_beat']} {\_num(wpb,2)}"

if wpb\_delta\_pct is not None:

arrow = ARROWS["up"] if wpb\_delta\_pct >= 0 else ARROWS["down"]

wpb\_piece += f" ({arrow}{\_num(abs(wpb\_delta\_pct),0)}%)"

parts.append(wpb\_piece)

if trend is not None:

arrow = ARROWS["up"] if trend >= 0 else ARROWS["down"]

parts.append(f"· {L['trend']} {arrow}{\_num(abs(trend),0)}%")

comment = \_trim(" ".join(parts), STRAVA\_COMMENT\_MAX)

# Lang beskrivelse (topp‑linje ≤140 + valgfri body)

header\_parts = []

if cgs is not None: header\_parts.append(f"{L['cgs']} {int(round(cgs))}")

if intensity is not None and duration is not None and quality is not None:

header\_parts.append(f"– {L['how\_hard']}:{int(round(intensity))} | {L['how\_long']}:{int(round(duration))} | {L['how\_even']}:{int(round(quality))}.")

header\_parts2 = []

if vi is not None: header\_parts2.append(f"{L['vi']} {\_num(vi,2)}")

if pa is not None: header\_parts2.append(f"{L['pa\_hr']} {\_num(pa,1)}%")

if wpb is not None:

wpb\_ex = f"{L['w\_per\_beat']} {\_num(wpb,2)}"

if wpb\_delta\_pct is not None:

arrow = ARROWS["up"] if wpb\_delta\_pct >= 0 else ARROWS["down"]

wpb\_ex += f" ({arrow}{\_num(abs(wpb\_delta\_pct),0)}%)."

header\_parts2.append(wpb\_ex)

desc\_header = \_trim(" ".join(header\_parts + ["·"] + header\_parts2), STRAVA\_DESC\_HEADER\_MAX)

# Body valgfri (kan utvides senere)

desc\_body = None

return PublishPieces(comment=comment, desc\_header=desc\_header, desc\_body=desc\_body)

**DoD 4.1**

* Bygget uten feil.
* build\_publish\_texts() returnerer tekst selv ved manglende VI/Pa:Hr/W/beat.
* Trimming bekreftet.

**Commit:** feat(m7): strava publish formatter (comment/description, lang, trim & fallbacks)

**4.2 CLI‑integrasjon med flagg**

**Fil:** cli/analyze.py (eller tilsvarende)

* Legg til nye flagg: --publish-to-strava, --dry-run, --lang no|en.
* Etter at SessionReport er skrevet, generer PublishPieces.
* Hvis --dry-run: print tekstene og returnér.
* Hvis ikke: finn activity\_id og kall Strava API.

**Aktivitets‑ID (siste importerte)**

* **Løsning A (anbefalt):** Når du kjører import i M6, lagre state/last\_import.json:

json

KopierRediger

{ "last\_activity\_id": 1234567890123456789, "imported\_at": "2025-08-12T21:10:00Z" }

* **Fallback:** Hvis state/last\_import.json ikke finnes, spør Strava om siste aktivitet for brukeren og ta den ferskeste.

**Kode‑skisse:**

python

KopierRediger

import json, os, sys

from formatters.strava\_publish import build\_publish\_texts

from strava\_client import StravaClient # se 4.3

def load\_last\_activity\_id():

p = "state/last\_import.json"

if os.path.exists(p):

with open(p,"r",encoding="utf-8") as f:

return json.load(f).get("last\_activity\_id")

return None

def publish\_to\_strava\_if\_requested(args, report):

if not args.publish\_to\_strava and not args.dry\_run:

return

pieces = build\_publish\_texts(report, lang=(args.lang or "no"))

if args.dry\_run:

print("[DRY-RUN] COMMENT:", pieces.comment)

print("[DRY-RUN] DESC:", pieces.desc\_header)

return

activity\_id = load\_last\_activity\_id()

if not activity\_id:

# OPTIONAL: fetch latest via StravaClient

activity\_id = StravaClient().get\_latest\_activity\_id()

sc = StravaClient()

# Kommentar (kortlinje)

sc.create\_comment(activity\_id, pieces.comment)

# Beskrivelse (header)

sc.update\_description(activity\_id, pieces.desc\_header)

**DoD 4.2**

* --dry-run viser begge tekstene.
* Kjøring uten flagg endrer ingenting.
* --lang en gir engelske labels.

**Commit:** feat(m7): CLI flags for Strava publish (publish-to-strava, dry-run, lang)

**4.3 Strava‑klient (bruk eksisterende token‑håndtering)**

**Fil:** cli/strava\_client.py

* Gjenbruk M6 token/refresh‑logikk.
* Metoder:
  + get\_latest\_activity\_id() -> int
  + create\_comment(activity\_id: int, text: str) -> None
  + update\_description(activity\_id: int, header: str) -> None

**Skisse (HTTP‑nivå):**

python

KopierRediger

import requests, os

API = "https://www.strava.com/api/v3"

class StravaClient:

def \_\_init\_\_(self):

# Gjenbruk din eksisterende tokenmanager

self.token = self.\_ensure\_access\_token()

def \_ensure\_access\_token(self) -> str:

# return valid token (refresh if needed)

...

def \_h(self):

return {"Authorization": f"Bearer {self.token}"}

def get\_latest\_activity\_id(self):

r = requests.get(f"{API}/athlete/activities", headers=self.\_h(), params={"per\_page":1})

r.raise\_for\_status()

arr = r.json()

return arr[0]["id"] if arr else None

def create\_comment(self, activity\_id: int, text: str):

# Kort kommentarlinje

requests.post(f"{API}/activities/{activity\_id}/comments",

headers=self.\_h(), data={"text": text}).raise\_for\_status()

def update\_description(self, activity\_id: int, header: str):

# Oppdater aktivitetens description

requests.put(f"{API}/activities/{activity\_id}",

headers=self.\_h(), data={"description": header}).raise\_for\_status()

**DoD 4.3**

* Kommentar og beskrivelse oppdateres på en testaktivitet.
* Rate‑limit og 401/403/5xx bubbler men med hyggelig feilmelding (gjenbruk din feilhåndtering).
* Fallback til get\_latest\_activity\_id() funker hvis state mangler.

**Commit:** feat(m7): Strava client for comments & description update

**5) Robusthet & tester**

**5.1 Python‑tester for formatter/CLI**

**Filtre:** tests/test\_publish\_formatter.py

python

KopierRediger

from cli.formatters.strava\_publish import build\_publish\_texts

def \_mk(cgs=88, vi=1.11, pa=2.4, wpb=1.59, base=1.45, trend=3.4):

return {

"scores":{"cgs":cgs,"intensity":93,"duration":82,"quality":88},

"vi":vi, "pa\_hr\_pct":pa, "w\_per\_beat":wpb, "w\_per\_beat\_baseline":base,

"trend":{"cgs\_last3\_avg":85,"cgs\_delta\_vs\_last3":trend},

"if":0.92

}

def test\_comment\_trimming():

r = \_mk()

pieces = build\_publish\_texts(r, "no")

assert len(pieces.comment) <= 280

def test\_missing\_fields\_fallback():

r = \_mk(vi=None, pa=None, wpb=None, base=None, trend=None)

pieces = build\_publish\_texts(r, "en")

assert "CGS" in pieces.comment

assert "VI" not in pieces.comment

**DoD 5.1**

* pytest -q grønt.
* Tester dekker trimming, språk, manglende felter.

**Commit:** test(m7): formatter unit tests (trim/lang/fallbacks)

**5.2 Rust‑tester (kjerneberegninger)**

* **Fil:** core/src/lib.rs (eller tilsvarende) — legg til #[cfg(test)]‑modul med:
  + Unit‑tester for: NP, IF, VI, Pa:Hr, W/beat (syntetiske serier).
  + Edge‑cases: ingen watt, ingen HR, korte økter (<10/20/60), pauser, outliers.
* **Golden‑tester:**
  + **Mappe:** core/tests/golden/
  + 3 ekte økter som fixtures (CSV) + forventet JSON med toleranser ± små avvik.
  + Test kjører analyse og sammenligner felter (bruk approx for floats).

**DoD 5.2**

* cargo test grønt.
* Golden‑tests låses (commit fixtures).
* Perf‑guard: 2t @ 1Hz ≤ 200ms på din maskin (en enkel benchmark‑test med time limit).

**Commit:**

* test(m7): rust unit tests for NP/IF/VI/Pa:Hr/WpB + edges
* test(m7): golden tests for 3 real sessions with tolerances
* test(m7): perf guard ≤200ms for 2h@1Hz

**6) Dokumentasjon**

**6.1 docs/CGS\_v1.md**

**Innhold:**

* Formler: NP, IF, VI, Pa:Hr, W/beat + baseline (28d, ±25% varighet).
* CGS‑vekter (0.4/0.3/0.3), klokkeform for intensitet, stykkevise skalaer for kvalitet.
* Fallbacks: HR‑basert intensitet, default 50 hvis alt mangler.
* Badges‑regler.
* Ytelseskrav/antakelser.

**6.2 docs/cli\_usage.md**

* Eksempler:
  + Batch m/trend:  
    python analyze.py --input data/\*.csv --out output/ --cfg config.json --batch --with-trend
  + Auto‑FTP / set‑FTP:  
    --auto-ftp / --set-ftp 260
  + Formater:  
    --format json|csv|both
  + Publisering:  
    --publish-to-strava --lang en / --dry-run

**6.3 docs/strava\_publish.md**

* Hva publiseres (eksempler på kort/long).
* Språkvalg, trimming, fallbacks.
* Hvordan deaktivere (ikke bruk flagget / --dry-run for test).
* Krav: gyldig token, siste aktivitet eller state/last\_import.json.

**DoD 6**

* Tre dokumenter finnes, lesbare, konsise.
* Alle CLI‑eksempler testet manuelt.

**Commit:** docs(m7): CGS v1, CLI usage, Strava publish

**7) Comms & commits (sluttføring)**

Kjør rekkefølgen under (forutsatt at testene er grønne):

1. feat(m7): strava publish formatter (comment/description, lang, trim & fallbacks)
2. feat(m7): CLI flags for Strava publish (publish-to-strava, dry-run, lang)
3. feat(m7): Strava client for comments & description update
4. test(m7): formatter unit tests (trim/lang/fallbacks)
5. test(m7): rust unit tests for NP/IF/VI/Pa:Hr/WpB + edges
6. test(m7): golden tests for 3 real sessions with tolerances
7. test(m7): perf guard ≤200ms for 2h@1Hz
8. docs(m7): CGS v1, CLI usage, Strava publish
9. chore(m7): mark M7 complete in Masterplan

**Masterplan‑endring:**

* Sett “M7 – Analysefunksjoner (effektivitet, treningsscore)” til **Ferdig**.
* Legg til ny rad i statusoversikt (2025‑08‑13) med beskrivelsen du postet (CGS, baseline, badges, trend, CLI, API).

**Endelig DoD for M7**

* analyze.py støtter --publish-to-strava, --dry-run, --lang.
* Kommentar + beskrivelse publiseres på en nylig importert aktivitet.
* Formatter håndterer manglende felter og lengdegrenser.
* pytest og cargo test grønne (inkl. golden + perf‑guard).
* docs/\*.md på plass og oppdatert.
* Masterplan oppdatert til **Ferdig**.

Gi beskjed hvis du vil at jeg skal produsere en “diff‑klar” Masterplan.md‑snutt eller fylle inn konkrete toleranser/skalapunkter i CGS\_v1.md med tall (jeg kan spytte ut ferdige tabeller direkte).

Spør ChatGPT