# New Repository Plan: Health‑Lab 2.0 (Beard‑Excluded, Nutrition & Cardio Added)

## 1 Purpose and Scope

This repository will be a **personal health laboratory** for longitudinal self‑experimentation. Building upon the previous Health‑Lab plan, it consolidates scalp, facial‑skin, body‑skin, hypertrophy and strength modules while **excluding the beard domain**, and adds new domains for **nutrition**, **cardio**, **sleep** and **time tracking**. The objectives are to:

* Provide a unified REST API (FastAPI) for recording health events, nutrition logs, cardio sessions, sleep data and time‑usage logs.
* Offer a mobile client (React Native) so the user can log routines, meals and workouts, view dashboards and receive schedule prompts.
* Track outcomes (hair density, TEWL, muscle mass), nutrition intake, cardiovascular fitness, sleep quality and time allocation in a time‑series database with analytics dashboards.
* Implement a scheduling engine and blackout rules to coordinate protocols across domains, including workout scheduling, rest periods and nutrition timing.
* Include comprehensive documentation for every protocol, product and outcome metric; maintain a data dictionary and variable definitions.

## 2 Architecture Overview

### 2.1 Domain‑Driven Modular Monolith

The project will follow Domain‑Driven Design (DDD). Each domain (e.g., scalp, nutrition, cardio) encapsulates its own models, services and APIs, keeping global code minimal[[1]](https://www.actidoo.com/en/blog/python-fastapi-domain-driven-design#:~:text=We%20usually%20try%20to%20move,principle%20is%20equivalent%20to%20data). For every domain we will define:

* **Models (models.py)** – ORM entities and Pydantic schemas for events and measurements.
* **Services (service.py)** – Business logic, validation and rule enforcement.
* **Domain API (domain\_api.py)** – Interfaces for cross‑domain interactions (e.g., retrieving nutrition totals or cardio readiness).
* **Web API (router.py)** – FastAPI routers for external clients.

### 2.2 Vertical Monorepo

The server and client will live in one vertical monorepo. A shared Python package will hold Pydantic models used by both the FastAPI server and the React Native app. Vertical monorepos are efficient because server and client are released simultaneously and share data schemas[[2]](https://sqr-075.lsst.io/#:~:text=note%20proposes%20a%20new%20vertical,from%20the%20same%20Git%20repository).

### 2.3 Directory Structure

healthlab/  
├── apps/  
│ ├── api/ # FastAPI application (server)  
│ │ ├── main.py  
│ │ ├── config.py  
│ │ ├── domains/  
│ │ │ ├── scalp/  
│ │ │ ├── facial\_skin/  
│ │ │ ├── body\_skin/  
│ │ │ ├── hypertrophy/  
│ │ │ ├── strength/  
│ │ │ ├── nutrition/ # new domain  
│ │ │ ├── cardio/ # new domain  
│ │ │ ├── sleep/ # new domain  
│ │ │ └── time\_tracking/ # new domain  
│ │ ├── scheduling/ # scheduling engine and blackout rules  
│ │ └── automations/ # Celery or AWS Lambda tasks  
│ └── mobile/ # React Native (Expo) app  
├── packages/  
│ └── domain\_core/ # shared Pydantic models and enums  
├── docs/ # comprehensive domain specs and architecture docs  
├── tests/ # unit and integration tests  
├── infrastructure/ # Terraform/CDK scripts for RDS, S3, ECS etc.  
└── pyproject.toml # build metadata for shared package

## 3 Domain Modules

### 3.1 Scalp (scalp)

Unchanged from the previous plan: port the scalp documentation, implement models for ProductBatch, Event types (wash, microneedle, etc.) and measurement tables (hair mass index, shed tests)[[3]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_hair_care_project.md#L65-L73). Implement blackout rules (e.g., 24 h after microneedling) and risk escalation triggers[[4]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_hair_care_project.md#L52-L60).

### 3.2 Facial Skin (facial\_skin)

Port the facial‑skin documentation with TEWL‑based blackout rules[[5]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_skin_care_project.md#L55-L64). Implement event types (wash, vitamin C serum, moisturizer, TAAN, etc.), measurement tables (TEWL, sebum, lesion counts)[[6]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_skin_care_project.md#L70-L77), and rules such as barrier guard.

### 3.3 Body Skin (body\_skin)

Create a body\_skin\_project.md similar to the facial‑skin doc. Define event types (wash, moisturize, exfoliate, retinol), measurement tables (TEWL, dryness score, keratosis pilaris index) and blackout rules (e.g., 48 h after retinol). Implement services to adjust frequency based on dryness metrics.

### 3.4 Hypertrophy (hypertrophy)

Port the hypertrophy plan: implement models for WorkoutEvent (exercise name, sets, reps, load, RIR, tempo), measurement tables (body composition, muscle thickness, strength tests). Build services for progressive overload and deload scheduling.

### 3.5 Strength (strength)

Port the strength plan: implement models for LiftEvent (squat, bench, deadlift, etc.), measurement tables (one‑rep max, velocity profiles, injury reports). Build periodization logic and injury‑based blackout triggers.

### 3.6 Nutrition (nutrition)

A new domain for tracking food intake and dietary protocols.

* **Documentation**: Create nutrition\_project.md describing how to log meals, compute nutrient intake, and support protocols (e.g., ketogenic, high‑protein cycles). Include a formulary of supplements and dietary interventions.
* **Data sources**: Use external food databases to enrich logs. The **FoodData Central (FDC) API** provides REST access to nutrient data; it exposes endpoints for food search (/foods/search) and food details (/food/{fdcId}) and is intended for developers to incorporate nutrient data[[7]](https://fdc.nal.usda.gov/api-guide/#:~:text=The%20FoodData%20Central%20API%20provides,into%20their%20applications%20or%20websites). Another option is the **Open Food Facts API**, which provides ingredients and nutritional values; it is open data that can be reused for any purpose[[8]](https://openfoodfacts.github.io/openfoodfacts-server/api/#:~:text=Open%20Food%20Facts%20is%20a,reuse%20it%20for%20any%20purpose). These APIs will be wrapped by a service to parse user‑entered food descriptions and return macro/micronutrient values.
* **Models**:
* FoodEntry: fields include id, timestamp, description (free‑text meal description), quantity, energy\_kcal, protein\_g, fat\_g, carbohydrate\_g, fiber\_g, and a nested micronutrients dict (vitamins, minerals). Additional metadata such as brand or barcode can be stored.
* SupplementEvent: events for supplements (e.g., creatine) with fields for dose and product.
* **Services & Rules**: Implement parsing of free‑text logs via external APIs, compute daily macro totals, compare against dietary targets, and generate alerts if certain micronutrients or calories are out of range. Support journaling of special diets (e.g., high‑protein days) and integrate with the scheduling engine to time meals relative to workouts.
* **APIs**: Provide endpoints to record meals, scan barcodes (if supported by the external API) and retrieve nutritional summaries over time.

### 3.7 Cardio (cardio)

A new domain dedicated to cardiovascular training and fitness.

* **Documentation**: Write cardio\_project.md describing different cardio protocols (steady‑state runs, interval sessions, cycling, swimming). Include guidance on heart‑rate zones and metrics for assessing cardiovascular fitness.
* **Models**:
* CardioSession: fields for id, start\_time, end\_time, activity\_type (running, cycling, rowing, swimming), distance\_km, duration\_min, average\_hr, max\_hr, calories\_burned, rpe (rate of perceived exertion), and optional notes.
* CardioMeasurement: periodic measurements of cardiovascular health such as resting\_hr, max\_hr\_estimate, vo2\_max, aerobic\_threshold, recovery\_hr. These metrics are informed by exercise science: **resting heart rate** (RHR) is the heart beats per minute at rest and lower RHR implies better cardiovascular efficiency[[9]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=Cardiovascular%20fitness%2C%20often%20referred%20to,you%20achieve%20your%20fitness%20goals); **maximum heart rate** (MHR) is the highest heart rate achievable during intense exercise and is often estimated as 220 minus age[[10]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=,MHR); **heart rate reserve** (HRR) is the difference between MHR and RHR and helps customise training zones[[11]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=,HRR); **VO₂ max** represents the maximum oxygen consumption and is considered the gold standard for assessing cardiovascular fitness[[12]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=); the **aerobic threshold** marks the point where oxygen demand exceeds supply, training below it improves endurance[[13]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=); **recovery heart rate** measures how quickly the heart returns to resting level after exercise and is a proxy for fitness[[14]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=); **time to exhaustion** measures endurance capacity[[15]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=).
* **Services & Rules**: Calculate training zones (Zone 1‑5) based on HRR and VO₂ max; schedule cardio sessions in coordination with hypertrophy/strength workouts; adjust intensity based on recovery markers (e.g., if RHR is elevated, reduce intensity). Provide progressive overload logic for cardio (e.g., gradually increasing duration or intensity). Use measurement trends to modify training plans.
* **APIs**: Endpoints to log sessions, retrieve personal cardio metrics, and view heart‑rate zone summaries.

### 3.8 Sleep (sleep)

A domain for sleep tracking and recovery, integrated with wearable data.

* **Documentation**: Write sleep\_project.md covering sleep physiology, recovery protocols, and how to interpret metrics. Explain how sleep intersects with training and nutrition (e.g., sleep debt reduces recovery capacity). Describe how to ingest data from WHOOP or other wearables.
* **Data ingestion**: WHOOP measures sleep quality by tracking four sleep stages—slow wave sleep, REM, light and awake—“with near‑perfect accuracy”[[16]](https://www.whoop.com/us/en/how-it-works/#:~:text=OPTIMIZE%20SLEEP%20TRAIN%20SMARTER%20RECOVER,STRESS%20BUILD%20ROUTINES%20MONITOR%20HEALTH). It calculates how much sleep you need based on recent patterns and strain, monitors wake events, efficiency and respiratory rate to optimize sleep performance[[17]](https://www.whoop.com/us/en/how-it-works/#:~:text=02%20Understand%20your%20sleep%20need). Use the WHOOP API (v2) or other wearables via OAuth 2.0 to retrieve nightly sleep sessions, sleep need, sleep performance and associated metrics (sleep stages, time in bed, disturbances). Data will be stored in a SleepSession model.
* **Models**:
* SleepSession: fields for id, start\_time, end\_time, duration\_min, stage\_durations (dictionary of light, REM, deep/SWS, awake durations), sleep\_need\_min, sleep\_efficiency, wake\_events, respiratory\_rate, resting\_hr, and hrv. Additional fields can capture WHOOP recovery score or sleep consistency.
* **Services & Rules**: Calculate sleep debt, schedule bedtime reminders, enforce rest days when recovery scores are low, and integrate sleep need into the scheduling engine. Provide algorithms to adjust training intensity based on sleep quality (e.g., postpone high‑intensity cardio if deep sleep was low).
* **APIs**: Endpoints to import sleep data (e.g., via OAuth), retrieve sleep trends, and generate sleep score reports.

### 3.9 Time Tracking (time\_tracking)

A domain for tracking how time is spent across activities to support productivity and wellness analysis.

* **Data sources**: Use manual logs and integrate with open‑source time‑tracking software. **ActivityWatch** is an open‑source, privacy‑first, cross‑platform time tracker that automatically records how you spend time on your devices[[18]](https://activitywatch.net/#:~:text=ActivityWatch%20is%20an%20app%20that,spend%20time%20on%20your%20devices). It tracks active applications and window titles out of the box and supports categorization[[19]](https://activitywatch.net/#:~:text=Features). Kimai is another option for structured project time tracking with a JSON API[[20]](https://www.kimai.org/#:~:text=Kimai%20is%20an%20open,even%20on%20your%20mobile%20device).
* **Models**:
* TimeEntry: fields for id, start\_time, end\_time, activity (e.g., research, work, exercise, household), category (e.g., cognitive, physical, personal), and optional device or application. The duration\_min can be computed. Additional tags (Pomodoro, deep work) can be added.
* TimeSummary: aggregated daily or weekly time per category.
* **Services & Rules**: Provide importers that parse logs exported by ActivityWatch or Kimai (e.g., JSON CSV) and reconcile them with manually logged offline activities. Generate daily time budgets and visualizations to understand how time is allocated across research, training, nutrition logging, etc. Use this data to adjust scheduling (e.g., ensure sufficient focus blocks for research and adequate rest periods). Encourage adherence to planned schedules by detecting overwork or underutilization.
* **APIs**: Endpoints to record manual time entries, import logs, and query aggregated time analytics.

## 4 Scheduling Engine and Blackout Management

The central scheduler will coordinate events across all domains. Its responsibilities include:

* **Rule enforcement**: Apply blackout constraints (e.g., scalp microneedle blackout, TAAN barrier guard) and rest periods; incorporate new rules such as cardio recovery (avoid intense cardio after heavy strength day) and sleep‑based adjustments.
* **Event generation**: Produce schedule templates for each domain (nutrition meal timing, cardio sessions, sleep reminders) and update them based on deload weeks, travel, missed sessions or time budget constraints.
* **Conflict resolution**: Detect overlaps (e.g., scheduled cardio session coincides with work meeting) and automatically reschedule or prompt user decisions. Use time‑tracking data to detect actual vs planned time usage and refine scheduling.
* **Outcome‑aware adjustments**: Adjust future events based on outcomes and metrics—e.g., high TEWL reduces retinol frequency, low VO₂ max prompts more Zone 2 cardio, inadequate sleep triggers lighter workout sessions.

The scheduler will run as a service within api/scheduling/ and use a task queue (Celery or RQ) for asynchronous job execution. It will expose endpoints for plan instantiation, modifications and queries. Persistent scheduled jobs will be stored in Postgres/TimescaleDB.

## 5 Measurement Pipeline and Analytics

* **Database**: Use **TimescaleDB** (PostgreSQL extension) for storing all time‑series data (events, measurements, nutrients, heart rates, sleep metrics, time logs). Continuous aggregates will compute rolling averages and summary statistics.
* **ETL processes**: Write ETL scripts to derive metrics such as hair density changes, TEWL trends, hypertrophy progress, nutrient intake over time, cardio fitness trajectories, sleep debt and time‑allocation profiles. ETLs will be triggered via Celery beat or AWS EventBridge.
* **Dashboards**: Configure Grafana dashboards for each domain: scalp/skin (HMI, TEWL), nutrition (macro/micronutrient intake), cardio (heart‑rate zone distribution, VO₂ max trends), hypertrophy and strength (progressive overload charts), sleep (stage durations, efficiency), and time tracking (time allocation by category). Provide high‑level health KPIs (e.g., correlation between sleep quality and strength gains).

## 6 Documentation

Every domain must have a master document in docs/ containing:

* **Product/formulary** (for nutrition: supplements and meal types; for cardio: exercise equipment; for sleep: devices and techniques).
* **Execution grid**: daily/weekly protocols with blackout windows and schedules for meals, workouts and sleep.
* **Risk & automation rules**: triggers for alerts (e.g., micronutrient deficiencies, elevated resting HR, prolonged sleep debt) and automated actions.
* **Measurement pipeline**: instruments, variables, precision targets and sample sizes; e.g., scalp HMI precision targets[[3]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_hair_care_project.md#L65-L73), TEWL measurement precision[[21]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_skin_care_project.md#L70-L76), cardio metrics (RHR measurement repeated at consistent times) and nutrition intake (quantification accuracy based on API data quality). Note that FoodData Central data are public but unsupervised; the user assumes risk[[22]](https://fdc.nal.usda.gov/api-guide/#:~:text=Central%20%20if%20a%20higher,request%20rate%20setting%20is%20needed).
* **Backend & Frontend integration**: directory paths and API endpoints.

The repository will use MkDocs to render documentation and auto‑generate API docs from FastAPI’s OpenAPI schema. A data dictionary will define each variable and its units.

## 7 Testing and Continuous Integration

* **Unit tests** for models, services and routers in each domain (including nutrition, cardio, sleep and time tracking).
* **Integration tests** for the scheduling engine and cross‑domain interactions (e.g., a day with cardio, hypertrophy and sleep logging; verifying nutrition totals and time budgets).
* **Import‑linter** contracts to enforce domain independence (e.g., nutrition cannot import hypertrophy code directly), as recommended in the DDD guide[[23]](https://www.actidoo.com/en/blog/python-fastapi-domain-driven-design#:~:text=Restriction%20of%20import%20between%20domains).
* **CI pipeline** using GitHub Actions to run tests, lints, import‑linter, build Docker images and deploy docs. Pre‑commit hooks will enforce code style.

## 8 Deployment and Infrastructure

* Use **Docker Compose** for local development (API server, TimescaleDB, Redis, Grafana, front‑end). Provide sample environment variables for FDC, Edamam or WHOOP API keys.
* Provide **Terraform/CDK** scripts in infrastructure/ to deploy production resources (AWS RDS, S3 for storage, ECS/EKS for container orchestration, EventBridge for scheduling, IAM roles). Configure secure secrets management for API keys and OAuth credentials.
* Adopt the vertical monorepo release strategy: publish the FastAPI server as a Docker image; release the shared domain models as a Python package; build the React Native app via Expo.

## 9 Roadmap and Milestones

1. **Repository scaffolding**: Set up directory structure, pyproject.toml and React Native project; configure CI, pre‑commit and import‑linter.
2. **Port existing domains (scalp, facial‑skin)**: Copy documentation; implement models/services/routers and measurement tables; build dashboards.
3. **Implement scheduling engine**: Develop generic schedule templates and blackout management; integrate scalp/skin routines and verify with tests.
4. **Add body‑skin domain**: Write body‑skin documentation; implement models, services and scheduling integration.
5. **Add hypertrophy and strength domains**: Implement training models, periodization logic and measurement pipelines; create analytics dashboards.
6. **Add nutrition domain**: Integrate FoodData Central or Edamam API; implement meal logging, nutrient computations, daily macro targets and dashboards; document dietary protocols.
7. **Add cardio domain**: Implement cardio session logging, cardio metrics models and training zones; integrate heart‑rate data via wearables (e.g., WHOOP or Apple Watch) and schedule runs/cycles; build dashboards.
8. **Add sleep domain**: Implement sleep data ingestion from WHOOP; store sleep sessions with stage durations, sleep need and efficiency; integrate recovery scores into scheduling; create sleep dashboards.
9. **Add time‑tracking domain**: Implement manual logging and importers for ActivityWatch/Kimai; compute time allocation metrics; integrate with scheduling engine; build dashboards.
10. **Iterative refinement**: Expand automation rules; add outcome‑aware adjustments; incorporate additional measurements (HRV, blood pressure); optimize cross‑domain analytics.

## Summary

The Health‑Lab 2.0 repository extends the original design by introducing **nutrition**, **cardio**, **sleep** and **time tracking** domains while continuing to exclude the beard module. Each new domain follows DDD principles and is encapsulated with its own models, services and APIs. Nutrition will integrate external food databases (FoodData Central, Open Food Facts) to compute macro and micronutrient intake[[7]](https://fdc.nal.usda.gov/api-guide/#:~:text=The%20FoodData%20Central%20API%20provides,into%20their%20applications%20or%20websites)[[8]](https://openfoodfacts.github.io/openfoodfacts-server/api/#:~:text=Open%20Food%20Facts%20is%20a,reuse%20it%20for%20any%20purpose). The cardio domain will track sessions and key cardiovascular metrics—resting heart rate, maximum heart rate, heart rate reserve, VO₂ max, aerobic threshold, recovery heart rate and time to exhaustion[[24]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=Cardiovascular%20fitness%2C%20often%20referred%20to,you%20achieve%20your%20fitness%20goals)—and schedule training accordingly. Sleep integration leverages WHOOP’s ability to monitor sleep stages and calculate sleep need and sleep efficiency[[25]](https://www.whoop.com/us/en/how-it-works/#:~:text=OPTIMIZE%20SLEEP%20TRAIN%20SMARTER%20RECOVER,STRESS%20BUILD%20ROUTINES%20MONITOR%20HEALTH), feeding recovery insights into the scheduler. A time‑tracking domain collects manual and automated logs (e.g., via ActivityWatch) to provide context on how time is allocated[[18]](https://activitywatch.net/#:~:text=ActivityWatch%20is%20an%20app%20that,spend%20time%20on%20your%20devices). The updated plan provides a comprehensive blueprint for building a holistic, data‑driven health laboratory with robust documentation, testing and deployment practices.

[[1]](https://www.actidoo.com/en/blog/python-fastapi-domain-driven-design#:~:text=We%20usually%20try%20to%20move,principle%20is%20equivalent%20to%20data) [[23]](https://www.actidoo.com/en/blog/python-fastapi-domain-driven-design#:~:text=Restriction%20of%20import%20between%20domains) Domain-driven design with Python and FastAPI | ActiDoo GmbH

<https://www.actidoo.com/en/blog/python-fastapi-domain-driven-design>

[[2]](https://sqr-075.lsst.io/#:~:text=note%20proposes%20a%20new%20vertical,from%20the%20same%20Git%20repository) A vertical monorepo architecture for FastAPI client-server codebases

<https://sqr-075.lsst.io/>

[[3]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_hair_care_project.md#L65-L73) [[4]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_hair_care_project.md#L52-L60) master\_hair\_care\_project.md

<https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_hair_care_project.md>

[[5]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_skin_care_project.md#L55-L64) [[6]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_skin_care_project.md#L70-L77) [[21]](https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_skin_care_project.md#L70-L76) master\_skin\_care\_project.md

<https://github.com/brandonkluzek/Health/blob/5e1ecbd553ec8fee2830d5a2d3bab315b002d358/docs/master_skin_care_project.md>

[[7]](https://fdc.nal.usda.gov/api-guide/#:~:text=The%20FoodData%20Central%20API%20provides,into%20their%20applications%20or%20websites) [[22]](https://fdc.nal.usda.gov/api-guide/#:~:text=Central%20%20if%20a%20higher,request%20rate%20setting%20is%20needed) API Guide | USDA FoodData Central

<https://fdc.nal.usda.gov/api-guide/>

[[8]](https://openfoodfacts.github.io/openfoodfacts-server/api/#:~:text=Open%20Food%20Facts%20is%20a,reuse%20it%20for%20any%20purpose) Introduction to Open Food Facts API documentation - Product Opener (Open Food Facts Server)

<https://openfoodfacts.github.io/openfoodfacts-server/api/>

[[9]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=Cardiovascular%20fitness%2C%20often%20referred%20to,you%20achieve%20your%20fitness%20goals) [[10]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=,MHR) [[11]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=,HRR) [[12]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=) [[13]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=) [[14]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=) [[15]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=) [[24]](https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones#:~:text=Cardiovascular%20fitness%2C%20often%20referred%20to,you%20achieve%20your%20fitness%20goals) Measuring Cardio Fitness: Understanding VO2 Max and Heart Rate Zones - Personal Training Blog

<https://www.westrive.com/blog/measuring-cardio-fitness-understanding-vo2-max-and-heart-rate-zones>

[[16]](https://www.whoop.com/us/en/how-it-works/#:~:text=OPTIMIZE%20SLEEP%20TRAIN%20SMARTER%20RECOVER,STRESS%20BUILD%20ROUTINES%20MONITOR%20HEALTH) [[17]](https://www.whoop.com/us/en/how-it-works/#:~:text=02%20Understand%20your%20sleep%20need) [[25]](https://www.whoop.com/us/en/how-it-works/#:~:text=OPTIMIZE%20SLEEP%20TRAIN%20SMARTER%20RECOVER,STRESS%20BUILD%20ROUTINES%20MONITOR%20HEALTH) How WHOOP Works | Health Monitoring, Sleep Tracking, Recovery Insights

<https://www.whoop.com/us/en/how-it-works/>

[[18]](https://activitywatch.net/#:~:text=ActivityWatch%20is%20an%20app%20that,spend%20time%20on%20your%20devices) [[19]](https://activitywatch.net/#:~:text=Features) ActivityWatch - Open-source time tracker

<https://activitywatch.net/>

[[20]](https://www.kimai.org/#:~:text=Kimai%20is%20an%20open,even%20on%20your%20mobile%20device) Kimai - Open Source time-tracker

<https://www.kimai.org/>