Comprehensive Feature Overview and Expansion Plan

Current Features of the Fitness App

- **User Accounts & Onboarding:** Users can sign up/in (with email and OAuth) and go through an onboarding process to set up their profile and fitness goals. This includes inputting personal data (age, weight, etc.) and fitness objectives, which helps personalize their experience.

- **Workout Programs & Tracking:** The app provides structured **workout routines** and an **exercise library**. Users can browse exercises (with details and instructions) and follow workout programs. As they complete workouts, the app tracks their sessions – capturing data like duration, calories burned, heart rate, distance (for cardio activities), and even GPS routes for outdoor workouts. Every workout session is logged with a type (strength training, running, cycling, etc.) and source (e.g. Apple Watch, Garmin, or manual entry). This allows the app to handle both gym workouts and cardio activities.

- **Muscle Heatmap & Progress Visualization:** A **muscle heatmap** feature visually shows which muscle groups have been engaged. After workouts, users can see a body diagram highlighting muscles used, helping them identify which areas they've been focusing on. Currently, the heatmap indicates muscle groups worked, and the **progress** section of the app likely includes charts or summaries of workout volume over time. Users can track personal records or improvements in weight lifted, reps, or pace. This helps in monitoring progress and ensuring a balanced training regimen.

- **Nutrition Tracking:** The app includes a nutrition tab for logging meals and tracking macros (proteins, carbs, fats). Users can record what they eat throughout the day (the UI shows meal icons like breakfast, lunch, dinner, snacks) and get a summary of their calorie and macronutrient intake. There may be a **photo logging or barcode scan** feature to quickly input foods (hinted by camera icons in the code). By comparing intake to goals, users can adjust their diet to stay on track. Basic nutrition guidance is provided to hit daily macro targets.

- **Recipes & Meal Plans (Premium):** In the premium section, users have access to a library of healthy **recipes** and possibly personalized meal plans. They can browse recipes aligned with their diet preferences and nutritional goals. The app might also suggest meal plans for goals like bulking, cutting, or general wellness. This provides actionable guidance on what to eat, not just tracking. *(Premium content encourages*

users to upgrade for more value.)

- **Supplement Guidance (Premium):** Another premium feature is information on **supplements**. Users can learn about vitamins, protein powders, etc., and see recommendations relevant to their goals (for example, suggesting whey protein for muscle gain, or electrolytes for endurance athletes). This educational content helps users optimize their nutrition and recovery.

- **Body Scanner & Metrics:** The app includes a **body scanner** feature (likely using the phone's camera or AR). This could allow users to scan their physique to track changes in body measurements or composition over time. For instance, it might estimate body fat percentage or simply record measurements (waist, biceps, etc.) using images. This ties into progress tracking, giving users another way to see improvements (beyond weight or strength). It's possibly a premium feature to add value for paid users.

- **Health & Recovery Tracking:** Beyond workouts, the app tracks daily health metrics to provide a holistic view of fitness. It records steps, distance moved, floors climbed, active vs. total calories, etc., each day. It also monitors heart rate data (resting HR, average, max, HRV) and sleep quality/duration if available. Users can see their sleep breakdown (light, REM, deep sleep, awake time) and a nightly sleep score out of 100. For users with advanced wearables, the app even captures **WHOOP strain and recovery scores**, as well as **Garmin stress level and body "Battery"** metrics. This indicates the app is capable of giving insight into recovery and readiness each day, not just workouts. All these data points feed into a **daily dashboard** on the Home tab, giving the user feedback on activity, recovery, and overall wellness.

- **Device Integrations (Apple, Google, WHOOP, Garmin):** A key feature is the ability to connect popular fitness devices and platforms so that data flows automatically into the app. Currently supported integrations include: **Apple Watch** (via HealthKit on iOS), **Google's Health Connect** for Android (covering Wear OS devices), **WHOOP** bands, and **Garmin** wearables. In the app's Profile > Devices section, users can link these accounts. Once connected, the app syncs data like workouts, heart rate, sleep, and more from the devices into the app's tracking system. For example, a run tracked on Garmin or a sleep session recorded by WHOOP will appear in the app without manual entry. This integration greatly enriches the data available to the user and ensures that **everything is connected to optimize their journey** (as you said) – the app becomes a central hub for all fitness and health data.

- **Goal Setting:** Users can set personal goals – whether it's a target weight, a weekly workout frequency, or specific strength milestones (like squatting X pounds). An **Edit Goals** interface allows updating these objectives. The app likely uses these goals to personalize recommendations (e.g., if the goal is weight loss, the nutrition targets and workouts suggested will align with cutting calories and increasing cardio).

- **AI Assistance (existing features):** The app already leverages some AI capabilities to enhance user experience. For instance, it can suggest **exercise alternatives** if a user wants to swap an exercise in a workout (using an AI service to recommend a similar exercise targeting the same muscles). There may also be an AI-powered chat or coach for basic tips. In the nutrition section, AI might help generate meal suggestions or analyze a user's diet. These AI features are relatively targeted right now – e.g., providing *technical guidance* on form or swapping exercises – but they lay the groundwork for deeper AI integration (as we'll discuss below).

- **User Profile & Insights:** Each user has a profile where personal data and preferences are stored. The profile also likely shows summary stats like total workouts completed, streaks, personal bests, etc. The **Insights/Progress** area might include charts of weekly training volume, body weight over time, or macro intake over time. This gives users feedback on how consistent they've been and highlights trends (for example, a chart of weekly calories burned or a calendar heatmap of workout days). Currently, the focus is on individual progress rather than social sharing – the app does not appear to have a friend system or community feed (it's more of a personal coach than a social network).

In summary, the current app is a **comprehensive fitness tracker and coach** with features spanning workouts, nutrition, and health metrics. It integrates with major devices to gather data and uses visual tools like muscle heatmaps and charts to inform the user's training. The inclusion of premium content (recipes, body scanning, supplements) suggests a freemium model where advanced guidance is available to subscribers.

Additional Features & Improvements to Consider

To **expand the app's capabilities** and truly "optimize your journey" with all data connected, here are several features and enhancements you could add:

- **Wider Device & App Integrations:** Extend support to *more wearables and fitness apps* beyond the current four integrations. For example:

    - **Fitbit:** Integrate the Fitbit API so users with Fitbit trackers can sync their steps, heart rate, sleep, and workouts. Fitbit provides a robust Web API for developers to access user data , making this integration feasible.

    - **Apple Health (iPhone):** In addition to Apple Watch, allow reading data directly from the iPhone's Apple Health app. This covers data sources like the phone's step counter or other apps linked to Apple Health. (HealthKit integration largely covers this, but ensure the app requests and utilizes all relevant HealthKit data – e.g., workouts recorded by third-party iPhone apps or Bluetooth devices.)

    - **Google Fit:** Although Google is moving toward Health Connect, adding direct Google Fit support can help users who still rely on it. This ensures even non-Wear OS Android users (or those whose devices sync to Google Fit) are covered.

    - **Fitbit Alternatives:** Consider other popular wearables such as **Oura Ring** (which offers an API to share its sleep and recovery data), **Polar** watches (Polar Flow API), and **Samsung Health** (many Samsung users track fitness there; Samsung Health can sync through Health Connect as well).

    - **Nike Run Club / Nike Training Club:** Nike's apps are popular for running and workouts. However, Nike does **not offer a public API** for direct integration . To support Nike data, one workaround is to integrate **Strava**. Nike's apps can sync activities to Strava (Nike's run data can be pushed to Strava, which *does* have a public API) . By integrating with the Strava API, the app can import runs, rides and other activities that users record on platforms like Nike Run Club, Peloton, etc. Strava's API is free to use and provides access to a wealth of workout data (segments, routes, etc.) with user permission .

    - **Other Fitness Apps:** Support importing data from **Nike Training Club**, **MyFitnessPal**, or **Google's upcoming apps**, if available. Even if direct integration isn't possible, allow users to **export or import** common data formats (GPX files for runs, CSV for workout logs, etc.) so they can bring their history into the app.

- *Technical note:* Each new integration will require using the platform's OAuth for authentication and their specific APIs for data. This can be complex, but there are services that simplify this. For example, tools like **Terra** or **Rook** provide unified APIs to connect to many wearables at once, so you integrate with one SDK and gain access to multiple device data streams . Whether you build it in-house or use such services, expanding integrations will make the app a one-stop hub for any user, regardless of what device or app they use to track fitness.

- **Expanded Workout Library (All Sports Support):** Currently the app is heavily focused on gym-style workouts (strength training, cardio). You want it to support *"every single kind of sport"* – which means broadening content and tracking for sports like football, soccer, basketball, hockey, swimming, skiing, **and more**. Key enhancements here:

    - **Exercise/Drill Database for Sports:** Add sport-specific drills and exercises. For example, for basketball include jump training, dribbling drills, agility ladder workouts; for soccer/football include sprint drills, footwork exercises, etc. Each sport could have a section with training plans and exercises tailored to skills and conditioning needed in that sport.

    - **Training Plans by Sport:** Create templates or programs for different sports (e.g., a “Off-Season Soccer Conditioning Program” or “Pre-Season Hockey Strength Program”). These plans would mix gym work with sport-specific conditioning. Users could choose their sport as a focus, and the app then personalizes their workout suggestions to that sport.

    - **Activity Tracking for Sports:** Leverage wearable data to log actual sport play. For instance, if a user plays a 90-minute soccer match, allow them to log it (or detect it via a wearable) as an activity. The app can record the duration, approximate calories, and maybe distance run during the game (some advanced watches can track sport modes like soccer or skiing). Ensure the **workout type taxonomy** is broad – include categories like **Team Sports, Swimming, Winter Sports, etc.** – so any activity the user does can be captured. This will reflect in their logs and health metrics (e.g., a skiing day might show high calorie burn and leg muscle usage).

    - **Specialized Metrics:** Incorporate metrics relevant to specific sports. For example, swimming laps and pace, cycling power or cadence, running VO2 max or split times, etc., if data is available from devices. For skiing/snowboarding, track altitude and slope difficulty if possible. These details will make the app more valuable to serious athletes in those sports.

    - **Injury Prevention & Flexibility:** Sports often come with injury risks, so include features like **stretching routines, mobility exercises, and prehab workouts** specific to each sport (e.g., shoulder mobility for swimmers, knee stability exercises for skiers). This again broadens the workout content and shows that the app is tuned to each athlete’s needs.

- By supporting all sports, the app transitions from a pure gym tracker to a comprehensive **sports performance platform**. A user could use it equally for weightlifting and for tracking their marathon training or soccer games. All that data would feed into their profile, giving a truly complete picture of their fitness journey.

- **Enhanced Heatmap & Analytics:** Upgrading the muscle heatmap and overall analytics will provide deeper insight:

    - **Intensity-Based Muscle Heatmap:** Evolve the heatmap to not just show which muscles were used, but *how much* they were taxed over a period. For example, use color gradients (light red for lightly worked, bright red for heavily worked) to indicate intensity or volume of training for each muscle group. This could be calculated based on the sets, reps, and weights lifted for that muscle (or time spent, for cardio-focused muscles). Users could toggle the heatmap to view today’s workout vs. the past week or month. This answers questions like “which

muscles have I been overusing or neglecting lately?" at a glance.

    - **Weekly/Monthly Training Load Charts:** Introduce charts that aggregate training load. For instance, a graph of **weekly workout intensity** (combining volume of weight lifted and cardio load) so users can see spikes or drops in their training. Another idea is a **calendar view** highlighting which days had workouts (a "workout streak" calendar) and color-coded by intensity or category (e.g., green for a cardio day, blue for strength day, etc.).

    - **Recovery and Readiness Trends:** If we have all that health data (resting HR, HRV, sleep, etc.), we can display a **recovery trend**. For example, a chart of the user's daily recovery score (from WHOOP or a custom algorithm) alongside their workout load. This can teach users to balance hard training days with rest. It could even alert them if their metrics suggest fatigue (e.g., "your resting heart rate is up and sleep was down this week – consider a deload or recovery day").

    - **Personal Records & Achievements:** Expand the progress section to celebrate personal bests (fastest 5K run, heaviest deadlift, longest swim, etc.). Include achievements or badges for milestones (like "100 workouts completed" or "Ran 50 miles total"). This gamification keeps users engaged and motivated to improve.

- Overall, these analytics improvements turn raw data into actionable feedback. An upgraded heatmap and new charts will make the app feel like an **expert training analyst** – highlighting patterns in training that the user might miss and suggesting how to optimize (e.g., "Your upper body is getting a lot more work than your legs – consider adding a leg session.⚠️").

- **Personalized Performance Scoring System:** Introduce an **"overall score"** and per-muscle/group scores to give users a simple quantitative summary of their fitness status:

    - **Overall Fitness Score:** This could be a composite index (0–100 or 0–10 scale, or even a 5-star rating) that rolls up various aspects: training consistency, workout intensity, recovery, and maybe progress toward goals. For example, someone who works out frequently, achieves their targets, and recovers well might score high, whereas inconsistent training or poor recovery would lower the score. This single number makes it easy for the user to gauge their journey at a glance.

    - **Muscle Group Scores:** As you described, assign each major muscle group a score out of 5 (or out of 10/100). This would represent how developed or well-trained that muscle is relative to the user's potential or relative to other muscles. It's somewhat analogous to what Fitbod calls a Muscle Strength Score

– Fitbod generates a score for each muscle group based on your training data and strength levels . In our case, a 5-star system might be easier to understand (e.g., Legs: 4/5, Chest: 3/5, Back: 5/5). These scores could be derived from the volume and progression in exercises targeting those muscles. If a muscle hasn't been trained much or is lagging in strength, its score would be lower.

    - **Insights and Improvement Tips:** For each muscle score, provide a brief analysis like "🌟🌟🌟☆☆ **Shoulders: 3/5.** You have decent shoulder strength, but there's room to improve. Consider adding overhead presses to your routine." This gives context. The app can detect imbalances – e.g. "Your quads are stronger than your hamstrings – include more hamstring exercises to avoid injury." It essentially turns the raw workout data into coaching advice.

    - **Progress Over Time:** These scores should update as the user trains. The app could show a trend (e.g., last month your shoulders were 2/5, now 3/5 – improvement!). This encourages users to follow the app's guidance. As Fitbod noted when introducing their muscle scoring, it lets users **see how strength changes over time for each muscle and identify weakest parts of the body to focus on** .

- By quantifying performance this way, the app gives users a clear target for improvement. It's much like a game where you try to level up each muscle group. It also simplifies complexity – instead of drowning in numbers (weights, reps, HRV, etc.), a user can just see "Legs: 5/5 – great, Arms: 2/5 – needs work" and intuitively know what to do.

- **"Improve" Button – AI Coach & Plan Overhaul:** This is a big one – an **AI-driven personal coach** that can overhaul workouts and diet when the user asks for improvement. Here's how it could work and the features involved:

    - **AI Assessment:** In a dedicated section (say **Coach** or **Analysis**), the user can tap "Assess My Overall Fitness." The app's AI will then analyze all the user's data – workouts, consistency, muscle scores, diet, and health metrics – and produce an **Overall Report**. This report would include the overall score and muscle scores above, plus narrative feedback on their progress ("You've been very consistent with workouts, great job! Your endurance has improved, evidenced by a lower resting heart rate. However, your upper body strength isn't increasing – likely because you're not increasing your training volume there.").

    - **Improvement Recommendations:** Within this report, the AI coach provides concrete suggestions: e.g., "To improve your chest (currently 3/5), we recommend adding 2 chest exercises per week. Also, your protein intake is a bit low for muscle gain – try to consume 20g more protein per day." The advice should cover **both workout adjustments and diet tweaks**, since you want it to

overhaul “workout and diet and everything.”

- **“Overhaul My Plan” Action:** If the user clicks a button like “Improve My Plan” or “Generate New Plan,” the app will **create a revamped workout program and meal plan** automatically. This is where generative AI or intelligent algorithms come in:
  - The new **Workout Plan** would account for the user’s goals, fix the weaknesses identified, and keep what’s working. For example, it might add an extra leg day if legs were undertrained, or include more cardio if the goal is endurance. It could also introduce new exercises (maybe ones the user hasn’t done before) to keep things fresh or address neglected muscles. This is similar to having a personal trainer redesign your program when you plateau.
  - The new **Diet Plan** would align with the workout changes and the user’s nutritional needs. Perhaps the AI increases the daily calorie target if muscle gain is desired, or swaps some foods to ensure enough protein and micronutrients. It could even lay out a sample daily meal schedule or suggest recipes from the library that fit the revised macros.
- **AI Personal Trainer Interaction:** The user could potentially chat or Q&A with this AI coach. For example, ask “Why did my plan change?” and the AI might respond “You indicated you want to improve your back strength, so I added more rowing exercises and increased your protein intake to support muscle growth.” This makes the feature interactive and educational.

- Under the hood, implementing this would involve machine learning or rule-based systems. Some apps already move in this direction – for instance, **FitnessAI** uses AI based on millions of workouts to optimize sets/reps for strength training , and **Strongr Fastr** generates meal plans from hundreds of recipes to hit your macro goals . Additionally, **Zing Coach** creates tailored workout plans in real-time and even does a body composition analysis via smartphone camera . These examples show that AI can dynamically adjust training and diet plans to a user’s data. By integrating a similar capability, your app would essentially offer each user a **personal trainer + nutritionist in their pocket**. The moment they want to “level up” their routine, the AI can deliver an updated plan to push them closer to their goals.

- **Comprehensive Sport-Specific Mode:** Since the app will cater to all sports, consider adding modes or settings specific to those sports:
  - For example, a **“Football Mode”** might allow a user to log practice drills, record game statistics (goals, assists, etc.), and get football-specific tips. A **“Swimming Mode”** could interface with swim trackers to log laps, and present specialized

metrics like SWOLF (a metric swimmers use). These modes might also change the dashboard emphasis – an endurance athlete might care more about VO2 max and weekly mileage, whereas a weightlifter cares about 1-rep max strength. The app can adapt what it displays based on the user's primary sport focus.

    - Sport communities often value comparisons and rankings. While a full social network is a big feature, you could incorporate small competitive or social aspects: e.g., showing how a user's stats compare to an average for their age/sport, or allowing them to share a summary of their game or a personal record to social media.

- Essentially, this is about **speaking the athlete's language**. If someone is a runner, the app should feel like a running coach; if they're a powerlifter, it should feel like a lifting coach. That personalization can set your app apart from one-size-fits-all fitness apps.

- **Technical Guidance & Implementation Notes:** (Since you mentioned wanting technical guidance, here are a few points on implementing these features):

    - **APIs & Data Management:** When connecting to external APIs (Fitbit, Strava, etc.), you'll need to register your app with those services to get API keys and use OAuth 2.0 flows for user authorization. Make sure to securely store tokens (as your current setup does in the database) and refresh them when needed. Leverage your existing Supabase DB schema (it already has tables for device connections and health metrics) to store the new data. For example, Fitbit data can be mapped to your daily_health_metrics (steps, calories, sleep, etc.) and workout sessions tables. Strava workouts (runs/rides) can be inserted as workout sessions with GPS routes. Reusing the standardized structure you have will make integration smoother.

    - **AI Coach Development:** For generating workout/diet plans, you could integrate a cloud AI service (like OpenAI's GPT-4 or another) and feed it the user's data and goals to get a plan. Alternatively, develop your own algorithm based on established training principles (this gives more predictable results). A hybrid approach could work: use your algorithm for the structured program, and an AI model to generate the natural-language explanations and tips. Start simple – e.g., a rule: "if muscle X score < 3/5, then add one more exercise for muscle X per week" – and gradually incorporate more variables.

    - **User Experience:** Introduce these features gradually and with user education. For instance, when the new scoring system rolls out, include a little tooltip explaining what the "Muscle Score" means. When the AI generates a new plan, perhaps present a summary for user approval ("We've created a new 4-week plan for you focusing on endurance – does this look good?") so they feel in control. Always allow customization – users might want to tweak the AI's plan

(maybe swap an exercise or a meal). Embracing that will make the AI feature feel like a helpful assistant, not a strict boss.

    - **Performance Considerations:** Some features like advanced analytics (weekly charts, AI computations) can be heavy. Use background tasks to crunch data (e.g., compute the weekly muscle scores or training load in the background, so it's ready when user opens the app). Caching results and updating them when new data comes in will keep the app responsive.

- With thoughtful implementation, these technical additions will function smoothly and provide a rich, data-driven experience to the user.

By adding the features above, the app would cover **virtually every aspect of a user's fitness journey**. It would track their daily activity, guide their workouts for any sport, monitor their recovery, advise their nutrition, and continuously adapt to their progress. All their devices and apps would feed into it, and all the guidance they need comes out of it – truly optimizing their fitness journey with a unified, smart platform.

Conclusion

The current app already has a strong foundation with workout tracking, nutrition, and device integrations. The proposed additions – from expanded device support (Fitbit, etc.) to sport-specific training and AI-driven coaching – will transform it from a general fitness app into an **all-in-one personal training ecosystem**.

Imagine a user experience where **everything is connected**: they go for a run with their Garmin, play a game of basketball, hit the gym for strength training, log their healthy meals, and wear a sleep tracker at night – and *one app* aggregates all that, analyzes it, and tells them **exactly** how to improve. That's the end goal. Each feature we add brings us closer: more data coming in from integrations, and smarter guidance going out via analytics and AI.

With an upgraded heatmap showing their muscle usage, a personal score card for their body, and an AI that can recalibrate their plan on demand, users will be empowered to continuously progress. Whether they are a casual gym-goer or an athlete in any sport, the app will adapt to them. This level of personalization and connectivity is what will set the app apart from alternatives like Nike's apps or others – essentially offering the convenience of a unified dashboard **plus** the expertise of a coach and nutritionist.

All these features align with the vision of optimizing one's fitness journey through data and intelligent guidance. By implementing them, you'll create a platform that not only tracks everything but also **makes sense of everything** for the user – guiding them to be their fittest, healthiest self with every tool and insight at their disposal. 🚀

Sources:

- App's Supabase schema indicating support for Apple HealthKit, Google Health Connect (Android), WHOOP, and Garmin integrations, and storing rich health metrics (steps, heart rate, sleep, strain, etc.).

- App's workout logging design capturing workout type, device source, distance and GPS route for activities (enabling tracking of runs, rides, etc. in addition to gym workouts).

- Fitbit Developer API documentation – confirms availability of a Web API for accessing Fitbit device data .

- Oura Ring API reference – demonstrates that third-party apps can retrieve Oura wearable data via API .

- Nike/Strava integration notes – Nike's lack of a public API and the ability to sync Nike Run Club data to Strava, which does offer an open API ; Strava's API overview .

- Rook API (example of aggregator) – one API to connect data from hundreds of wearables .

- Fitbod's "Muscle Strength Score" – example of scoring each muscle group's development on a relative scale to highlight strengths/weaknesses .

- AI Fitness Coaching examples: *Strongr Fastr* (AI diet planning) , *FitnessAI* (AI-generated strength workouts) , and *Zing Coach* (real-time AI plan adjustments + body scan) , which illustrate the kind of AI-driven plan optimization proposed.

Before you start (applies to all OAuth providers)

- **Redirect URL pattern** (use one per provider):

  https://<your-app-domain>/api/oauth/callback/<provider>

  (Examples below use /api/oauth/callback/<provider>—match whatever your Rork backend/router expects.)

- **Store secrets safely**: Put Client IDs/Secrets in **Rork env vars** (and **never** in client code).

- **Token storage**: Save access_token, refresh_token, expires_at, user_id/provider_user_id, scopes in Supabase (table: oauth_tokens).
- **Sync job**: Create a background job (cron/queue) that:
    1. refreshes tokens when close to expiry
    2. pulls latest data (since last sync)
    3. upserts into your normalized tables (see “Unified data model”, at the end).

---

1) Apple Health / Apple Watch (HealthKit – iOS native)

No OAuth (data is on device). You request permissions on-device and write a small native plugin/bridge (Rork supports custom native modules or use Capacitor/React Native bridge).

Steps

1. **Apple Developer**
    - Enroll in Apple Developer Program.
    - In **Identifiers > App**, enable **HealthKit** capability.
2. **Xcode / iOS project**
    - Add **HealthKit** capability to the iOS target.
    - In Info.plist, add:
        - NSHealthShareUsageDescription (why you read health data)
        - NSHealthUpdateUsageDescription (if you write)
    - If you’re hybrid (RN/Capacitor), install a HealthKit bridge (e.g., RN Healthkit or custom).

3. **Request permissions at runtime**
    - Choose data types you need (workouts, heart rate, steps, calories, sleep, VO2Max, body mass, etc.).
    - Call HealthKit authorization (read types; write optional).
4. **Read data**
    - On app open and periodically (or user pull-to-refresh), query:
        - **Workouts** (type + duration + energy + distance)
        - **Quantity types** (steps, heart rate samples, calories)
        - **Category types** (sleep analysis)
    - Normalize and save to Supabase.
5. **Testing**
    - Use **Health** app > Sources to confirm your app has access.
    - Manually add test workouts in Health app or simulate with Apple Watch.

Gotchas

- Apple rejects apps that read excessive health data without clear user benefit. Ask only for what you use.
- Data never leaves device unless **you** upload it—be explicit in privacy policy.

---

2) Google Health (Android) – Health Connect

No OAuth. It's a device-level permission (like HealthKit), and surfaces data from Google Fit/Wear OS and partner apps.

Steps

1. **Google Play Console**
    - Set app up; target Android 13+ recommended for Health Connect.
2. **Android project**
    - Add Health Connect dependency.
    - Request permissions (steps, distance, heart rate, sleep, workouts, calories, VO2Max if available).
3. **Runtime flow**
    - On first connect, request Health Connect permissions via the SDK.
    - Query data periodically (workouts, samples) and push to Supabase.
4. **Testing**
    - Install **Health Connect** on the device (if not preinstalled).
    - Use a Wear OS watch or a partner app (e.g., Fitbit on Android) that writes into Health Connect.

Gotchas

- Health Connect is sandboxed per app—always check granted permissions.
- Some metrics are aggregated; some are sample-level (heart rate). Handle both.

---

3) Strava (to cover Nike Run Club indirectly)

OAuth 2.0. NRC can sync to Strava; you then pull from Strava.

Steps

1. **Create Strava App**
    - Go to Strava Developer Portal → Create App.
    - Set **Authorization Callback Domain** to your domain.
    - Get **Client ID** and **Client Secret**.
2. **Scopes**
    - read (basic), activity:read (and activity:read_all if you need private activities).
    - Add offline_access if Strava provides refresh tokens for your app tier (it does).
3. **OAuth flow**
    - Send user to Strava authorize URL with scopes.
    - Receive code at /api/oauth/callback/strava.
    - Exchange for access_token, refresh_token, expires_at.
4. **Data sync**
    - Endpoints: Athlete, Activities (list & detail), Streams (GPS, HR).
    - Pull activities since last_synced_at, upsert workouts and samples (route, pace, HR if consented).
5. **Testing**
    - Connect NRC → Strava in the Strava mobile app (user action).
    - Record a run in NRC; verify it appears via your Strava API.

Gotchas

- Rate limits exist—batch sync and backoff.

- Private activities require activity:read_all.

---

4) Oura Ring

OAuth 2.0 (or PAT in some flows). Great for sleep, HRV, readiness.

Steps

1. **Oura Cloud Developer**
    - Create an app; obtain **Client ID/Secret**.
    - Set redirect: /api/oauth/callback/oura.
2. **Scopes**
    - daily (readiness, activity), heartrate, sleep, workout (as available).
3. **OAuth flow**
    - Standard code exchange → tokens + expiry.
4. **Data sync**
    - Pull **sleep sessions**, **readiness**, **activity**, **HR/HRV**.
    - Map to your tables: sleep_sessions, readiness_scores, daily_metrics.
5. **Testing**
    - Wear overnight; verify last night's sleep imports.

Gotchas

- Oura timestamps are local → convert to UTC and store user timezone.

---

5) Fitbit

OAuth 2.0. Broad consumer base; steps, HR, sleep, workouts.

Steps

1. **Fitbit Developer**
    - Register app → get **Client ID/Secret**.
    - Redirect: /api/oauth/callback/fitbit.
2. **Scopes**
    - activity, heartrate, sleep, profile, nutrition (if you want), weight.
    - Request offline_access for refresh tokens.
3. **OAuth flow**
    - Standard code → tokens.
    - Fitbit requires exact redirect URI match (HTTPS).
4. **Data sync**
    - Pull daily summaries (steps, calories, distance), **intraday HR** (premium scope access may apply), **sleep**, **activities**.
    - For intraday HR/minute-level you may need extra permission in app settings.
5. **Testing**
    - Do a walk with Fitbit; check steps & HR import.

Gotchas

- Intraday HR has additional approvals/rate limits—start with daily aggregates if needed.
- Units (imperial/metric) differ per user—normalize.

---

6) Garmin

OAuth 2.0, but **Garmin Health API requires approval/contract**. If you can't get Health API access, you have two options:

- Use **Garmin Connect Developer Program** (partners) with webhooks, **or**
- Let users sync Garmin → Strava → your app (quickest path).

Direct Garmin Steps (if approved)

1. **Apply to Garmin Health**
    - Once approved, create an app; obtain **Client ID/Secret**.
    - Set redirect: /api/oauth/callback/garmin.
2. **Scopes**
    - Activity, Heart Rate, Sleep, Stress/Body Battery (if your plan includes).
3. **OAuth flow**
    - Standard code → tokens.
4. **Webhooks (preferred)**
    - Configure data push endpoints (activity completed, sleep ready).
    - Verify webhooks; handle retries.
5. **Data sync**
    - Backfill via pull APIs; then stay current via webhooks.

Gotchas

- Approval can take time. If blocked, enable **Garmin → Strava** path for users.

---

7) WHOOP

OAuth 2.0. Key metrics: **Strain, Recovery, Sleep**.

Steps

1. **WHOOP Developer**
    - Create a client; get **Client ID/Secret**.
    - Redirect: /api/oauth/callback/whoop.
2. **Scopes**
    - Typically: read:recovery, read:cycles, read:workout, read:sleep, plus offline_access.
3. **OAuth flow**
    - Standard code → tokens.
4. **Data sync**
    - Pull **cycle** (day) summaries, **recovery scores**, **strain**, **sleep stages**, **workouts**.
    - Map to daily_readiness, sleep_sessions, workouts, hrv/resting_hr.
5. **Testing**
    - Check morning recovery publishes next day; verify timezone alignment.

Gotchas

- WHOOP “days” are cycle-based (not midnight-to-midnight). Respect their window when computing daily scores.

Unified data model (so “everything is connected”)

Create/extend these Supabase tables so all providers fit cleanly:

- **profiles**: user_id, dob, sex, height_cm, weight_kg, primary_sport, tz
- **oauth_tokens**: user_id, provider, access_token, refresh_token, expires_at, scopes, provider_user_id
- **workouts**: id, user_id, provider, type (run, ride, strength, swim, sport_*), start_at, end_at, duration_s, distance_m, energy_kcal, avg_hr, max_hr, route_geojson
- **workout_sets** (for strength): workout_id, exercise_id, muscle_groups[], weight_kg, reps, rir
- **sleep_sessions**: id, user_id, provider, start_at, end_at, duration_s, stages_json, score
- **daily_metrics**: user_id, date, steps, active_kcal, resting_kcal, floors, hrv_ms, resting_hr, vo2max, strain, recovery, readiness, stress, body_battery
- **permissions_log**: user_id, provider, granted_scopes, granted_at

Your **heat map** & **muscle scores** read from workout_sets (volume by muscle group) and your **overall score** blends daily_metrics + adherence (number of completed planned workouts / week).

Frontend “Connect” flows (copy)

Add a **Devices** page with buttons:

- **Apple Health (iOS)** → “Enable” → HealthKit permission sheet
- **Google Health (Android)** → “Enable” → Health Connect permission sheet
- **Strava / Oura / Fitbit / Garmin / WHOOP** → “Connect” → open provider OAuth → land on /api/oauth/callback/<provider> → show “Connected ✓”

Background sync schedule (baseline)

- **On connect**: import **last 30–90 days**.
- **Every 15 min**: check for near-expiry tokens and refresh.

- **Hourly**: pull new **workouts** & **samples** (Strava/WHOOP/Oura/Fitbit/Garmin).
- **Daily at 06:00 local**: recompute **overall score**, **muscle scores**, and **heat map**.

Security & privacy checklist

- Minimize scopes (ask only what you use).
- Encrypt tokens at rest; rotate secrets.
- Clear **consent** and **privacy policy** pages (GDPR/CCPA).
- Offer “Disconnect” per provider (revoke tokens + delete data on request).

What Rork needs from you (per provider)

- **Strava**: Client ID, Secret, Redirect URL
- **Oura**: Client ID, Secret, Redirect URL
- **Fitbit**: Client ID, Secret, Redirect URL
- **Garmin**: Client ID, Secret, Redirect URL (or confirm Strava fallback)
- **WHOOP**: Client ID, Secret, Redirect URL
- **Apple**: HealthKit enabled in bundle + usage strings
- **Google (Android)**: Health Connect permissions in manifest + onboarding copy

---

What’s inside

1) .env template

Includes all keys/placeholders you'll need:

- Supabase (SUPABASE_URL, SUPABASE_ANON_KEY, SUPABASE_SERVICE_ROLE_KEY)
- OAuth for Strava, Oura, Fitbit, Garmin, WHOOP (with proper callback URLs)
- Flags for Apple HealthKit / Google Health Connect (no secrets needed)
- Web3Forms access key, support email/WhatsApp
- Optional: OpenAI, Stripe, Firebase/FCM, Analytics, Sentry, security secrets

Replace your-app-domain.com with your actual domain and paste your real client IDs/secrets.

2) Supabase SQL schema

A unified schema so "everything is connected," covering:

- profiles, oauth_tokens
- workouts, workout_sets, exercises
- sleep_sessions, daily_metrics, permissions_log
- muscle_volume_cache (for fast heatmap rendering by 7d/30d)
- Indexes, triggers, and **RLS policies** so each user can only see/change their own data

Quick hookup notes

- **Redirect URIs** (keep this pattern):
    - https://<your-domain>/api/oauth/callback/strava
    - https://<your-domain>/api/oauth/callback/oura
    - https://<your-domain>/api/oauth/callback/fitbit
    - https://<your-domain>/api/oauth/callback/garmin
    - https://<your-domain>/api/oauth/callback/whoop

- **Apple HealthKit / Google Health Connect**: no OAuth; request device permissions in-app and then read data locally, push to Supabase.

- **Background sync**: after connecting, backfill 30–90 days, then run hourly sync + daily recalculations (overall score, muscle heatmap).