**Strava Webhook Automation System**

**Technical Documentation & Data Flow Analysis**

**Executive Summary**

We successfully implemented an automated data synchronization system that captures Strava activity data in real-time and stores it in a Supabase database. This eliminates the need for manual data syncing and provides immediate access to activity metrics for analysis and tracking purposes.

**System Architecture Overview**

**Core Components:**

1. **Strava API & Webhook System** - Source of activity data and event notifications
2. **Supabase Edge Function** - Webhook handler and data processor
3. **Supabase PostgreSQL Database** - Data storage and management
4. **Authentication & Token Management** - OAuth flow and token refresh handling

**Detailed Data Flow**

**1. Initial Setup & Registration Phase**

**Systems Involved:**

* Strava API (https://www.strava.com/api/v3/)
* Supabase Edge Functions (https://jmyqirpxiyxfwxpisyhu.supabase.co/functions/v1/strava-webhook)
* Local registration script (register\_webhook.py)

**Workflow:**

1. **Webhook Registration**: Python script registers our Edge Function endpoint with Strava
2. **Verification Challenge**: Strava sends GET request with verification parameters
3. **Challenge Response**: Edge Function validates token and returns challenge response
4. **Subscription Confirmed**: Strava confirms webhook subscription (ID: 296264)

[register\_webhook.py] → [Strava API] → [Edge Function] → [Verification Success]

**2. Real-Time Activity Sync Workflow**

**Trigger Event:** Athlete uploads new activity or edits existing activity in Strava mobile/web app

**Systems & Data Flow:**

**Step 1: Webhook Event Notification**

* **System**: Strava Webhook Service
* **Action**: Sends POST request to Edge Function with event metadata
* **Data**: {object\_type: "activity", object\_id: 12345, aspect\_type: "create", owner\_id: 67890, ...}

**Step 2: Event Processing & Authentication**

* **System**: Supabase Edge Function (strava-webhook)
* **Actions**:
  + Validates event is activity creation/update
  + Queries athletes table for user's access token
  + Checks token expiration (5-minute buffer)
  + Refreshes OAuth token if needed using refresh token

SELECT access\_token, refresh\_token, expires\_at

FROM athletes

WHERE id = {owner\_id}

**Step 3: Strava API Data Retrieval**

* **System**: Strava Activities API
* **Action**: Fetches detailed activity data using access token
* **Endpoint**: GET /api/v3/activities/{activity\_id}
* **Data Retrieved**:
  + Basic metrics (distance, time, speed)
  + Heart rate data (average, max)
  + Power data (watts, kilojoules)
  + Geographic and elevation data
  + Activity metadata (name, description, sport type)

**Step 4: Database Storage**

* **System**: Supabase PostgreSQL Database
* **Tables Updated**:

**activities Table:**

INSERT INTO activities (

id, athlete\_id, name, sport\_type, start\_date,

distance, moving\_time, elapsed\_time,

total\_elevation\_gain, average\_heartrate, max\_heartrate,

average\_speed, max\_speed, average\_watts, kilojoules,

description

) VALUES (...)

**Step 5: Heart Rate Zone Processing (If Available)**

* **System**: Strava Zones API + Supabase Database
* **Action**: Fetches heart rate zone distribution data
* **Endpoint**: GET /api/v3/activities/{activity\_id}/zones
* **Storage**: Saves time spent in each HR zone to heart\_rate\_zones table

INSERT INTO heart\_rate\_zones (

activity\_id, zone\_1\_time, zone\_2\_time,

zone\_3\_time, zone\_4\_time, zone\_5\_time

) VALUES (...)

**Technical Implementation Details**

**Authentication & Security**

* **OAuth 2.0 Flow**: Standard Strava authentication with access/refresh tokens
* **Token Management**: Automatic refresh before expiration (5-min buffer)
* **Webhook Verification**: Secure token-based verification for webhook authenticity
* **Database Security**: Service role key for bypassing RLS in Edge Function

**Error Handling & Resilience**

* **Network Failures**: Graceful handling of Strava API timeouts
* **Token Expiration**: Automatic refresh with fallback error handling
* **Missing Data**: Null value handling for optional activity metrics
* **Database Conflicts**: Upsert operations prevent duplicate entries

**Configuration Management**

* **Environment Variables**: Secure storage of API credentials via Supabase secrets
* **Edge Function Config**: JWT verification disabled for public webhook access
* **Import Maps**: Proper Deno module resolution for TypeScript compatibility

**Database Schema**

**athletes Table**

- id (primary key) - Strava athlete ID

- username - Strava username

- firstname, lastname - Athlete name

- access\_token - OAuth access token (encrypted)

- refresh\_token - OAuth refresh token (encrypted)

- expires\_at - Token expiration timestamp

**activities Table**

- id (primary key) - Strava activity ID

- athlete\_id (foreign key) - Links to athletes table

- name - Activity title

- sport\_type - Activity type (Run, Ride, Swim, etc.)

- start\_date - Activity start timestamp

- distance - Distance in meters

- moving\_time, elapsed\_time - Duration in seconds

- total\_elevation\_gain - Elevation in meters

- average\_heartrate, max\_heartrate - HR in BPM

- average\_speed, max\_speed - Speed in m/s

- average\_watts, kilojoules - Power metrics

- description - Activity notes

**heart\_rate\_zones Table**

- activity\_id (foreign key) - Links to activities table

- zone\_1\_time through zone\_5\_time - Time in each HR zone (seconds)

**Key Achievements & Benefits**

**Automation Accomplished:**

✅ **Real-Time Sync**: Activities appear in database within seconds of Strava upload  
✅ **Zero Manual Intervention**: No more clicking sync buttons or running scripts  
✅ **Comprehensive Data Capture**: Full activity metrics including heart rate zones  
✅ **Multi-Athlete Support**: System scales to handle multiple connected athletes  
✅ **Token Management**: Automatic OAuth token refresh prevents authentication failures

**Technical Benefits:**

* **Serverless Architecture**: Edge Functions scale automatically with usage
* **Cost Efficient**: Pay-per-invocation model, no idle server costs
* **High Availability**: Supabase infrastructure provides 99.9% uptime
* **Real-Time Analytics**: Data immediately available for dashboards and reporting

**Business Value:**

* **Immediate Data Access**: Activity data available instantly for analysis
* **Reduced Maintenance**: No manual sync processes to manage
* **Scalable Solution**: Handles growing number of athletes and activities
* **Data Consistency**: Single source of truth for all activity metrics

**System Monitoring & Maintenance**

**Monitoring Points:**

* **Edge Function Logs**: Real-time webhook processing status
* **Database Metrics**: Activity insert/update success rates
* **Token Refresh Events**: OAuth token management health
* **Webhook Subscription Status**: Strava webhook registration health

**Maintenance Tasks:**

* **Quarterly**: Review webhook subscription status with Strava
* **Monthly**: Monitor database storage growth and optimize if needed
* **Weekly**: Check Edge Function error rates and performance metrics
* **As Needed**: Update Strava API integration for new features

**Deployment Configuration**

**Supabase Edge Function Setup:**

[functions.strava-webhook]

enabled = true

verify\_jwt = false # Required for public webhook access

entrypoint = "./functions/strava-webhook/index.ts"

**Environment Variables:**

STRAVA\_CLIENT\_ID=66224

STRAVA\_CLIENT\_SECRET=f3e67948cf...

STRAVA\_VERIFY\_TOKEN=05978704df8c945ee89a3eca83453cc540595530

# SUPABASE\_URL and SUPABASE\_SERVICE\_ROLE\_KEY automatically provided

**Webhook Registration:**

Webhook ID: 296264

Callback URL: https://jmyqirpxiyxfwxpisyhu.supabase.co/functions/v1/strava-webhook

Status: Active and verified

**Success Metrics**

The implementation achieved 100% of the stated objectives:

1. ✅ **Automated Data Flow**: Strava → Webhook → Database (no manual steps)
2. ✅ **Real-Time Processing**: Activity data available within seconds
3. ✅ **Complete Data Capture**: All relevant activity metrics stored
4. ✅ **Reliable Operation**: Successful webhook verification and data sync
5. ✅ **Production Ready**: Error handling, logging, and monitoring in place

This system transforms manual, batch-oriented data collection into a seamless, real-time automated workflow that scales with usage and requires minimal ongoing maintenance.