API DOCUMENTATION SAMPLE

FamNest API Reference

Five production REST endpoints, documented from the live codebase.

System	FamNest — AI wellness coach for parents (Next.js 16 · Supabase · Groq Llama 3.3 70B)
Scope	Authenticated user endpoints, payment webhook, and scheduled jobs
Author	Virginia Mwega — Technical Writer & Documentation Engineer
Source	Extracted directly from `app/api/**/route.ts`, June 2026

1. Overview

FamNest exposes a small set of server-side REST endpoints under `/app/api` in addition to its primary server-action data flow. This document covers every HTTP-reachable route in the production codebase: what it does, how it authenticates, and what it returns — written by reading the implementation, not by guessing at intent.

Endpoints fall into three categories:

- **User-facing** — called from the browser by a signed-in user (`/api/export`).
- **Payment** — called by Flutterwave, the payment processor (`/api/webhooks/flutterwave` , `/api/billing/callback`).
- **Scheduled** — called by Vercel Cron on a fixed schedule, never by a browser (`/api/cron/*`).

2. Authentication patterns

Pattern	Used by	How it works
Supabase session cookie	`/api/export`	Request is authenticated via the user's existing Supabase session cookie. The route rejects with `401` if `supabase.auth.getUser()` returns no user.
Webhook signature header	`/api/webhooks/flutterwave`	Flutterwave sends a static secret in the `verif-hash` header. It is compared to `FLUTTERWAVE_SECRET_HASH` using `crypto.timingSafeEqual` to avoid timing attacks — never a plain `===` on secret values.
Bearer cron secret	`/api/cron/*`	Vercel Cron sends `Authorization: Bearer <CRON_SECRET>` automatically when `CRON_SECRET` is set. The route is otherwise unauthenticated, so this header is mandatory in production.
None (redirect flow)	`/api/billing/callback`	No auth — this is a public redirect target. It re-verifies the transaction server-side against the Flutterwave API before granting anything, so the query string itself is never trusted.

3. Endpoints

`GET` **`/api/export`** Session cookie required

Exports the signed-in user's full check-in history as CSV.

Query params	None
Rate limit	10 requests / hour / user, backed by a Postgres function (durable across cold starts — see §5)
Success	`200` — `text/csv` body, `Content-Disposition: attachment`
Errors	`401` not signed in · `429` rate limit exceeded

```
# Response headers
Content-Type: text/csv; charset=utf-8
Content-Disposition: attachment; filename="famnest-checkins-2026-06-24.csv"
Cache-Control: no-store

# Response body
date,mood,stress_level,sleep_hours,available_minutes,goal
2026-06-20T07:12:00.000Z,4,2,7.5,15,"Get through the morning without yelling"
```

POST `/api/webhooks/flutterwave`

verif-hash header

Receives payment events from Flutterwave. Handles both first purchase and auto-renewal — both arrive as the same `charge.completed` event type.

Required header	`verif-hash` — must constant-time-equal `FLUTTERWAVE_SECRET_HASH`
Body	Flutterwave event payload, e.g. `{ "event": "charge.completed", "data": { "id": 123, "status": "successful" } }`
Success	`200` — `{ "received": true }`
Errors	`401` bad/missing signature · `400` unparseable body · `500` handler error

> **Security note:** the webhook body is never trusted for the actual grant. On a valid `charge.completed` event the handler calls `verifyTransaction()` against the Flutterwave API directly before calling `activatePremiumFromTransaction()` — the webhook is a trigger to re-check, not a source of truth. Unrecognized events (e.g. `subscription.cancelled`) are acknowledged with `200` and ignored, since they're handled elsewhere in-app.

GET `/api/billing/callback`

Public — re-verified server-side

Where Flutterwave's hosted checkout redirects the browser after payment. Makes the first purchase feel instant instead of waiting on webhook delivery; the webhook above remains the source of truth for renewals.

Query params	`status` (string), `transaction_id` (string)
Behavior	Verifies the transaction server-side, then 303-redirects to `/billing` with a status flag
Redirects	`?checkout=success` · `?checkout=cancelled` · `?checkout=error`

GET `/api/cron/daily-reminder`

Bearer CRON_SECRET

Runs daily. Sends a reminder email to every user who hasn't checked in today and hasn't opted out.

Success	`200` with a per-run summary, e.g. `{ "totalUsers": 412, "skippedAlreadyCheckedIn": 188, "targeted": 224, "sent": 219, "failed": 5 }`
Errors	`401` missing/wrong bearer token · `500` on a Supabase query failure (returned as `{ "error": "<message>" }`)

GET `/api/cron/weekly-report`

Bearer CRON_SECRET

Runs weekly for Premium subscribers only. Pulls each user's last 7 days of check-ins, asks Groq (Llama 3.3 70B) for a short, kind summary, saves it, and emails it. Falls back to a templated (non-AI) summary if the Groq call fails on quota — a paying user never sees a broken report because of an upstream rate limit.

Eligibility filter	`subscriptions.plan IN (monthly, annual)` and `status IN (active, trialing)`
Success (no eligible users)	`200` — `{ "premiumUsers": 0, "sent": 0 }`
Errors	`401` missing/wrong bearer token · `500` on a Supabase query failure

4. Error format & status codes

FamNest's API does not use a single global error envelope — error shape follows the endpoint's purpose:

- **Plain-text body** for simple auth/rate-limit rejections (`401` , `429`) — e.g. `export` and the webhook.
- **JSON** `{ "error": "..." }` for query-level failures inside cron jobs, since their caller (Vercel) reads structured logs, not a UI.
- **Redirect with query flag** for the browser-facing billing callback, since the consumer is a page, not a script.

5. Rate limiting

Rate limiting is backed by a Postgres function (`check_rate_limit`), not an in-memory counter — an in-memory `Map` resets on every serverless cold start and would silently stop limiting. The limiter **fails open**: if the limiter RPC itself errors, the request is allowed. For a wellness app, never locking out a real parent due to an infrastructure hiccup matters more than perfectly enforcing a cap.
