1. **Executive Summary**

This updated blueprint shows how to build an **AI‑powered Applicant Tracking System (ATS)** that starts on **Supabase Cloud** for database, auth and vector search, and stores all CV files in **AWS S3**. You can migrate later to self‑hosted Supabase without rewriting code. The guide explains how the app works, lists every building block, and walks you through construction—including JWT auth and Stripe pay‑per‑job paywall.

1. **System‑at‑a‑Glance**

React / Retool UI ⇄ FastAPI + Supabase JS SDK

↓

Supabase (Postgres + pgvector, Auth, Realtime)

↓

AWS S3 (object storage for all CV PDFs)

↓

Supabase Edge Functions (or Python workers)

1. **Core Building Blocks**

| **Need** | **Technology** | **Why** |
| --- | --- | --- |
| Relational DB & Vector search | **Supabase Postgres + pgvector** | Single service handles structured data *and* similarity queries. ([supabase.com](https://supabase.com/docs/guides/database/extensions/pgvector?utm_source=chatgpt.com)) |
| Authentication | **Supabase Auth (JWT)** | E‑mail, magic link & social login; JWT integrates with Row‑Level Security. ([supabase.com](https://supabase.com/docs/guides/auth/jwts?utm_source=chatgpt.com), [supabase.com](https://supabase.com/docs/guides/auth?utm_source=chatgpt.com)) |
| Realtime updates | Supabase Realtime | UI can show “scoring finished” instantly. ([supabase.com](https://supabase.com/docs/guides/realtime?utm_source=chatgpt.com)) |
| Background jobs | **Supabase Edge Functions – Background Tasks** or Python Celery | Run parsing & embedding outside the main request. ([supabase.com](https://supabase.com/docs/guides/functions/background-tasks?utm_source=chatgpt.com)) |
| Object storage | **AWS S3** (S3 SDK) | Cheap, scalable; Supabase Storage is S3‑compatible so switching is trivial. ([supabase.com](https://supabase.com/blog/s3-compatible-storage?utm_source=chatgpt.com), [supabase.com](https://supabase.com/docs/guides/storage/s3/compatibility?utm_source=chatgpt.com)) |
| Résumé parsing | resume‑parser + **SkillNER** | Extract entities & skills into JSON. |
| Text embeddings | OpenAI text‑embedding‑3‑large (cloud) **or**pgvector local embedding | Turn CV & JD text into vectors; stored directly in Postgres. ([supabase.com](https://supabase.com/blog/openai-embeddings-postgres-vector?utm_source=chatgpt.com)) |
| Scoring formula | Python function in worker | Blends similarity, industry match, skills & experience. |
| Industry classifier | Lightcast NAICS API or zero‑shot Llama | Labels employer sector codes. |
| Payments | **Stripe Usage‑based Billing API** | Charges recruiters one credit per job post. ([docs.stripe.com](https://docs.stripe.com/billing/subscriptions/usage-based?utm_source=chatgpt.com)) |
| Self‑hosting path | Supabase Docker Compose | Same open‑source services; export DB & buckets, import locally. ([supabase.com](https://supabase.com/docs/guides/self-hosting/docker?utm_source=chatgpt.com), [supabase.com](https://supabase.com/docs/guides/self-hosting?utm_source=chatgpt.com)) |

1. **Flow Explained in Plain English**
2. **Recruiter opens a job** → React/Retool calls Supabase RPC create\_job, Stripe deducts a credit.
3. **Candidate uploads CV** → React form POSTs to FastAPI /apply; file streams into S3 and entry is inserted into Supabase candidates table with status pending.
4. **Edge Function parses** the PDF, extracts JSON, generates embeddings, stores vectors in pgvector column, classifies industries.
5. **Score & explain** → Function computes total\_score and calls GPT‑4o (or local Llama) for 3‑bullet explanation, writes back to candidate\_scores.
6. **Realtime event** notifies the recruiter dashboard; Appsmith table refreshes automatically.
7. **Download** → /report/export/:jobId cloud function streams an Excel file; another endpoint zips all S3 PDFs and returns a presigned link.
8. **Step‑by‑Step Build Guide**

**Phase 0 – Prereqs**

* Supabase Cloud project (Pro tier for pgvector)
* AWS S3 bucket ats-resumes
* Stripe account

**Phase 1 – Supabase schema**

create table jobs (

id uuid primary key default uuid\_generate\_v4(),

org\_id uuid references auth.users,

title text,

jd text,

s3\_folder text,

opens\_at timestamptz,

closes\_at timestamptz

);

create table candidates (

id uuid primary key default uuid\_generate\_v4(),

job\_id uuid references jobs,

resume\_url text,

status text default 'pending'

);

alter table candidates add column embedding vector(1536);

Enable Row‑Level Security so each org sees only its own rows.

**Phase 2 – Auth & Paywall**

1. Configure Supabase Auth with email magic links.
2. Stripe webhook → Edge Function updates org\_credits table.
3. Postgres RLS policy checks org\_credits.remaining > 0 before insert into jobs.

**Phase 3 – Edge Functions**

* /apply handles multipart upload → S3.
* Background task parses PDF & updates row.

**Phase 4 – React or Retool UI**

* Login widget (Supabase Auth JS).
* “New Job” form; “Candidate Table” with sort/filter.

**Phase 5 – Export & Download**

* Edge Function queries candidate\_scores, uses openpyxl to stream XLSX.
* Generates S3 presigned ZIP link for all CVs.

**Phase 6 – Move to self‑host later**

1. supabase db dump the cloud DB. ([supabase.com](https://supabase.com/blog/s3-compatible-storage?utm_source=chatgpt.com))
2. Spin up local Docker stack (supabase start). ([supabase.com](https://supabase.com/docs/guides/self-hosting/docker?utm_source=chatgpt.com))
3. psql restore; sync S3 files with rclone copy s3:ats-resumes selfhost:ats-resumes.