Setup Guide — Run the Handbook Generator (PDF → RAG → Chat)

1) Prerequisites

- **Python 3.10+** (3.11 recommended)
- A **Supabase** project (Postgres)
- Internet access (for embedding model download on first run)
- (Optional) **xAI API key** for Grok (`XAI_API_KEY`)

---

2) Install dependencies

Option A: Use `requirements.txt` (recommended)

Make sure your `requirements.txt` includes at least:

```
streamlit
python-dotenv
supabase
pdfplumber
sentence-transformers
openai
tqdm
numpy
torch>=2.2.0
transformers>=4.43.0
datasets
einops>=0.8.0
```

Then run:

```
pip install -r requirements.txt
```

Option B: Install manually

```
pip install streamlit python-dotenv supabase pdfplumber sentence-transformers openai tqdm
numpy "torch>=2.2.0" "transformers>=4.43.0" datasets "einops>=0.8.0"
```

---

3) Create `.env` file

Create a file named `.env` in the repo root:

```
SUPABASE_URL=YOUR_SUPABASE_URL
SUPABASE_SERVICE_KEY=YOUR_SUPABASE_SERVICE_KEY

# Optional: Grok via xAI (OpenAI-compatible)
XAI_API_KEY=YOUR_XAI_API_KEY
# or GROK_API_KEY=YOUR_XAI_API_KEY

GROK_MODEL=grok-4-1-fast-reasoning
GROK_MAX_TOKENS=4000
```

Where to find Supabase values:

- Supabase → **Project Settings** → **API**
    - Project URL → `SUPABASE_URL`
    - service_role key → `SUPABASE_SERVICE_KEY`
      ⚠️ Keep the service key private (never expose client-side).

---

4) Supabase setup (Required)

Open Supabase → **SQL Editor** and run these in order.

4.1 Enable pgvector

```
create extension if not exists vector;
```

4.2 Create tables

```
create table if not exists public.documents (
  id uuid primary key default gen_random_uuid(),
  filename text not null,
  created_at timestamptz default now()
);

create table if not exists public.chunks (
  id uuid primary key default gen_random_uuid(),
  document_id uuid references public.documents(id) on delete cascade,
  chunk_index integer not null,
  content text not null,
  metadata jsonb default '{}'::jsonb,
  embedding vector(384) not null
);

create index if not exists chunks_document_id_idx on public.chunks(document_id);
```

4.3 Create retrieval function (RPC)

```
create or replace function public.match_chunks(
  query_embedding vector,
  match_count integer,
  filter_doc uuid default null
)
returns table (
  id uuid,
  document_id uuid,
  chunk_index integer,
  content text,
  metadata jsonb,
  similarity double precision
)
language sql
stable
as $$
  select
    c.id,
    c.document_id,
    c.chunk_index,
    c.content,
    c.metadata,
    1 - (c.embedding <=> query_embedding) as similarity
  from public.chunks c
  where (filter_doc is null or c.document_id = filter_doc)
  order by c.embedding <=> query_embedding
  limit match_count;
$$;
```

---

5) Run the app

From the repo root:

streamlit run app/main.py

It will open in your browser.

---

6) How to use

6.1 Index a PDF

1. Upload a **text-based PDF** in the sidebar
2. Click **Index PDF**
3. You should see: `Indexed ✅ document_id=...`

6.2 Ask questions (RAG chat)

Ask something in the PDF:

- You should get an answer with citations like `(PDF p. 2)`
  Ask something not in the PDF:
- You should get: **“The uploaded PDFs don't mention this.”**

6.3 Generate a handbook

In chat:

/handbook Retrieval-Augmented Generation

Download the output from the sidebar as Markdown.

---

7) Troubleshooting

PowerShell blocks venv activation

Run:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

Or activate via cmd:

```
venv\Scripts\activate.bat
```

Retrieval returns no results

Confirm chunks exist:

```
select count(*) from chunks;
```

Confirm RPC works:

```
select * from match_chunks(
  (select embedding from chunks limit 1),
  5,
  (select document_id from chunks limit 1)
);
```

First run is slow

The embedding model downloads on the first run; later runs are much faster.