Backend Setup Manual (No Setup Script)

October 1, 2025

Who is this for?

This guide is for anyone who clones the repository and *does not* already have Postgres or Docker on their machine. It provides two setup paths:

- **Path A (Docker-first)**: Use Docker Desktop to run Postgres in a container (recommended for most).
- **Path B (Local install)**: Install PostgreSQL + pgAdmin 4 locally.

It also covers Node.js, Python 3.10, Ollama (LLM runtime), ffmpeg, and how to restore/share the database.

1 What you need (summary)

- **Node.js 18+**: `https://nodejs.org`
- **Python 3.10** + packages (see §6)
- **Ollama** (LLM server) + a model (`llama3:3b` or `llama3:8b`) (§8)
- **ffmpeg** for Whisper STT (§7)
- **EITHER** Docker Desktop (§2) **OR** local PostgreSQL + pgAdmin 4 (§3)
- **.env** config (§5)

2 Path A: Postgres via Docker (recommended)

Install Docker Desktop

1. Install Docker Desktop: `https://www.docker.com/products/docker-desktop`
2. (Windows) Enable WSL integration if prompted; otherwise defaults are fine.

Bring up Postgres with your data

We ship a plain SQL dump at `db_dump/smartdb.sql`. You can restore it into a Postgres container in two ways:

Option A1 — Compose (auto-restore on first start). Create a `docker-compose.yml` like this in the repo root:

```
# docker-compose.yml
services:
  db:
    image: postgres:17
    container_name: smart-pg
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: 211021
      POSTGRES_DB: smartdb
    volumes:
      # any *.sql here will run on first start (empty volume)
      - ./db_dump:/docker-entrypoint-initdb.d:ro
      - pgdata:/var/lib/postgresql/data
volumes:
  pgdata:
```

Then:

```
docker compose up -d
```

On the very first start (empty volume), Postgres will auto-run `db_dump/01_smartdb.sql` if present. You can rename your dump to `01_smartdb.sql` for clarity.

Option A2 — Manual restore into a running container. If you already have a container named `smart-pg`:

```
# copy and restore inside the container
docker cp db_dump/smartdb.sql smart-pg:/tmp/smartdb.sql
docker exec smart-pg sh -c "psql -U postgres -d smartdb -f /tmp/smartdb.sql"
```

(If `smartdb` does not exist yet, create it: `createdb -U postgres smartdb` inside the container or from host.)

Credentials to use

- Host: `localhost`
- Port: `5432`
- User: `postgres`
- Password: `211021` (from compose)
- Database: `smartdb`

3 Path B: Local PostgreSQL + pgAdmin 4

Install

1. Install PostgreSQL (Windows installer includes pgAdmin 4): `https://www.postgresql.org/download/`
2. Or install pgAdmin 4 separately: `https://www.pgadmin.org/download/`

Restore the provided dump

Open a terminal (PowerShell or bash) and run:

```
createdb -h localhost -p 5432 -U postgres smartdb
psql -h localhost -p 5432 -U postgres -d smartdb -f db_dump/smartdb.sql
```

GUI (pgAdmin 4)

1. Add new server: host `localhost`, port `5432`, user `postgres`, password `211021`.
2. Create DB `smartdb` if it doesn't exist.
3. Query Tool → open `db_dump/smartdb.sql` → **Run**.

4 Recreate the dump (for maintainers)

If you make changes and want to re-share the DB, generate a new dump.

A) Dump from inside the container (version-safe)

```
# dump to /tmp inside container
docker exec smart-pg sh -c "PGPASSWORD=211021 \
  pg_dump -U postgres -d smartdb -F p -f /tmp/smartdb.sql"

# copy to host
docker cp smart-pg:/tmp/smartdb.sql db_dump/smartdb.sql

# optional cleanup
docker exec smart-pg rm /tmp/smartdb.sql
```

B) Stream directly to a host file (PowerShell pipeline)

```
docker exec -t -e PGPASSWORD=211021 smart-pg `
  pg_dump -U postgres -d smartdb -F p |
  Set-Content -Encoding UTF8 .\db_dump\smartdb.sql
```

C) Matching client locally

If you run `pg_dump` on the host, ensure the client version matches the server major version (e.g., Postgres 17 dump with `pg_dump 17`).

Schema-only (no data):

```
pg_dump -U postgres -d smartdb -s > db_dump/schema.sql
```

Full cluster (roles + all DBs):

```
docker exec -t smart-pg pg_dumpall -U postgres > db_dump/cluster.sql
```

5 Node.js & Backend

Install dependencies

```
npm install
```

Environment file

Create `.env` in repo root:

```
DB_TYPE=postgres

# Postgres (Docker or local)
PG_USER=postgres
PG_PASSWORD=211021
PG_HOST=localhost
PG_PORT=5432
PG_DATABASE=smartdb
```

Security note: Do **not** commit real passwords. Commit a `.env.example` with placeholders and ask users to copy it to `.env`.

Start the server

```
node server2.mjs
```

Backend will listen on `http://localhost:3003`.

6 Python 3.10 & packages

Install Python 3.10 and packages (Windows with `py` launcher):

```
py -3.10 -m pip install --upgrade pip
py -3.10 -m pip install ^
  langchain langchain-community sentence-transformers faiss-cpu PyPDF2 pdfminer
      .six ^
  openai-whisper sounddevice scipy pyttsx3
```

(Use \ line breaks on Linux/macOS; ^ for Windows CMD.)

7 ffmpeg (for Whisper STT)

- Windows: `winget install -id Gyan.FFmpeg -e`
- Confirm in terminal: `ffmpeg -version`

8 Ollama (LLM runtime)

1. Install Ollama: `https://ollama.com/download`
2. Start Ollama (daemon at `http://localhost:11434`).
3. Pull a model:

   ```
   ollama pull llama3:3b
   # or
   ollama pull llama3:8b
   ```

4. The Python script `langchain_query.py` expects an Ollama server running locally. If your script pins `llama3:3b` but you pulled `8b`, align them for consistency.

9 Project folders

Create required folders if they don't exist:

```
mkdir knowledge_base
mkdir uploads
mkdir sessions
mkdir quiz_memory
mkdir generated
```

10 First test

Upload a PDF (build vector index)

```
curl -X POST http://localhost:3003/api/upload-pdf \
  -F "course=vorkurs_chemie" \
  -F "file=@C:\path\to\your.pdf"
```

Ask a question

```
curl -X POST http://localhost:3003/api/semantic-chat \
  -H "Content-Type: application/json" \
  -d "{\"course\":\"vorkurs_chemie\",\"message\":\"What is a buffer?\"}"
```

11 What users *must* have installed (clarified)

- **Always:**
  - Node.js 18+
  - Python 3.10 + the listed Python packages
  - ffmpeg
  - Ollama + a pulled model (e.g., `llama3:3b`)
- **Database:** Choose one:
  - **Docker Desktop** (then run Postgres in a container), **or**
  - **Local PostgreSQL server** (installed natively) + optionally pgAdmin 4.

Users do *not* need both Docker and local Postgres—**only one path** is required.

12 Troubleshooting

- **pg_dump version mismatch:** Dump using the same major version as the server; easiest is dumping *from inside the container* (§4A).
- **Cannot connect to Docker daemon (WSL):** Run commands from Windows PowerShell, or enable WSL integration in Docker Desktop, or call Windows `docker.exe` from WSL with full path.
- **Ollama not reachable:** Start Ollama, ensure `http://localhost:11434` is up, and that the model is pulled.
- **Permissions on Windows paths:** Prefer repo-relative paths (e.g., output audio to `./generated`) instead of user-profile hard paths.

13 FAQ

- **Do I need Docker installed?** Only if you choose Path A. If you install Postgres locally (Path B), you don't need Docker.
- **Do I need Postgres installed?** Not if you use Docker (Path A). Docker will run Postgres for you.
- **Do I need pgAdmin 4?** Optional. It's a GUI. You can use CLI tools only.
- **What about credentials/secrets?** Do not commit real passwords. Provide `.env.example` and ask users to copy to `.env`.