1) Reproducible Install and Running

I documented and validated two reproducible setup paths

- pip path
    - Create and activate a virtual environment
    - Install dependencies from requirements.txt
    - Install package in editable mode (pip install -e)
- uv path
    - Create and activate a virtual environment with uv venv
    - Sync exactly to requirements.txt using uv pip sync
    - Install editable package (pip install -e)

After installation, the Flask app runs with python src/app.py, and the analysis page routes load successfully. Tests run with pytest.

2) Dependency Graph Summary

The dependency graph centers on app.py, which serves as the Flask entry point and route layer. Then app.py coordinates loading and cleaning with load_data.py and query_data.py. Database configuration and connection logic are centralized in db.py, separating secret/config concerns from business logic. The module_2 package (scrape.py and clean.py) handles data acquisition and normalization before database insertion.

3) SQL Injection Defense

I refactored SQL execution to remove unsafe query construction patterns using user input. No SQL is built from f-strings, string concatenation, or .format() with raw user-controlled SQL text. Dynamic SQL uses psycopg composition primitives. Queries also include built-in LIMIT behavior, and limit values are clamped to a safe max range to reduce risk/

4) Least-Privilege Database Config

Database credentials are not hard-coded in source; the app reads them from environment variables (DB_HOST, DB_PORT, DB_NAME, DB_USER, DB_PASSWORD), with optional DATABASE_URL override. The .env.example is included for setup, and .env is gitignored to prevent committing secrets. The application role is configured as non-superuser, with no createdb/createrole/owner-level permissions. Only required privileges are granted for current app behavior: CONNECT on database, USAGE on schema, and table-level SELECT/INSERT on public.applicants; sequence usage is granted only as needed for insert IDs. Enforces least privilege by allowing only necessary operations and limiting access if anything is exposed.