That's an excellent question! Providing clear instructions for others (or your future self) to run the project locally is crucial. Here's a comprehensive README.md or setup instructions you can include in your project:

# AI Document Verification App

This Flask application processes and verifies documents (Purchase Orders and ATS/Resumes) using OCR, predefined rules, and database comparisons.

## Prerequisites

Before you begin, ensure you have the following installed on your local machine:

1. **Python:** Version 3.11 or 3.12 is recommended (as Python 3.13 might have issues with some dependencies like psycopg2-binary on Windows). You can download Python from [python.org](https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.python.org%2Fdownloads%2F).
2. **Git:** For cloning the repository. You can download Git from [git-scm.com](https://www.google.com/url?sa=E&q=https%3A%2F%2Fgit-scm.com%2Fdownloads).
3. **pip:** Python's package installer (usually comes with Python).
4. **(Optional but Recommended for Windows for psycopg2):**
   * **PostgreSQL Client Libraries (including pg\_config):** If you encounter issues installing psycopg2-binary directly, installing the full PostgreSQL client tools can help. Download from [PostgreSQL downloads](https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.enterprisedb.com%2Fdownloads%2Fpostgres-postgresql-downloads) and ensure "Command Line Tools" are included during installation. Then, add the PostgreSQL bin directory to your system's PATH.
   * **Microsoft C++ Build Tools:** Needed if psycopg2 needs to be compiled from source. Install "Build Tools for Visual Studio" from the [Visual Studio website](https://www.google.com/url?sa=E&q=https%3A%2F%2Fvisualstudio.microsoft.com%2Fdownloads%2F) (select the "Desktop development with C++" workload).

## Local Setup Instructions

1. **Clone the Repository:**  
   Open your terminal or command prompt and run:
2. git clone https://github.com/atghemanshu/po.git
3. cd po

1. **Create and Activate a Virtual Environment:**  
   It's highly recommended to use a virtual environment to manage project dependencies.
2. # Replace 'python' with 'python3', 'py -3.11', or 'python3.11' if needed,
3. # depending on your Python installation and desired version.
4. python -m venv venv

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

Activate the virtual environment:

* + **Windows (PowerShell/CMD):**
  + .\venv\Scripts\activate

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Powershell

IGNORE\_WHEN\_COPYING\_END

* + **macOS/Linux (Bash/Zsh):**
  + source venv/bin/activate

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

Your terminal prompt should now be prefixed with (venv).

1. **Install Dependencies:**  
   With the virtual environment activated, install the required Python packages:
2. python -m pip install --upgrade pip
3. pip install -r requirements.txt

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

* + **Troubleshooting psycopg2-binary on Windows:** If you encounter errors related to pg\_config during this step (even after ensuring the venv is active), it means pip couldn't find a pre-compiled binary for your Python/OS.
    - Ensure you are using Python 3.11 or 3.12.
    - If problems persist, you may need to install the PostgreSQL client libraries and C++ Build Tools as mentioned in the "Prerequisites" section.

1. **Set Up Environment Variables (.env file):**  
   This project uses a .env file to manage configuration and sensitive credentials.
   * In the root of the project directory (po/), create a file named .env.
   * Add the following content to your .env file, replacing the placeholder values with your actual Supabase credentials and a new secret key:
   * DATABASE\_URL="postgresql://postgres:[YOUR\_SUPABASE\_DB\_PASSWORD]@[YOUR\_SUPABASE\_PROJECT\_HOST]:5432/postgres"
   * SECRET\_KEY="generate\_a\_new\_strong\_random\_secret\_key\_here"
   * OCR\_SPACE\_API\_KEY="K87955728688957" # Replace with your actual key if you have one, or use the placeholder for limited use
   * ADMIN\_INITIAL\_PASSWORD="your\_desired\_initial\_admin\_password"

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Env

IGNORE\_WHEN\_COPYING\_END

* + - **DATABASE\_URL**: Get this from your Supabase project settings (Project Settings -> Database -> Connection string URI). Replace [YOUR\_SUPABASE\_DB\_PASSWORD] and [YOUR\_SUPABASE\_PROJECT\_HOST].
    - **SECRET\_KEY**: Generate a strong random string. You can use Python: python -c "import secrets; print(secrets.token\_hex(24))"
    - **OCR\_SPACE\_API\_KEY**: Your API key for OCR.space. The placeholder key K87955728688957 is a public test key with limitations.
    - **ADMIN\_INITIAL\_PASSWORD**: The password for the default admin user (admin@example.com) that will be created on the first run.

1. **Database Setup (Table Creation & Initial Admin User):**  
   The application is configured to create database tables and an initial admin user if they don't exist when you first run app.py.
   * Ensure your Supabase instance (referenced in DATABASE\_URL) is running and accessible.
   * The tables will be created based on the SQLAlchemy models defined in app.py.
   * The initial admin user will be:
     + Email: admin@example.com
     + Password: The value you set for ADMIN\_INITIAL\_PASSWORD in your .env file (or admin@a123 if the env var is not set and the app has that default).
2. **Run the Flask Application:**  
   With the virtual environment still active and the .env file configured:
3. python app.py

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

You should see output indicating that the Flask development server is running, typically on http://0.0.0.0:5000/ or http://127.0.0.1:5000/. The console will also show messages about table creation and admin user seeding on the first run.

1. **Access the Application:**  
   Open your web browser and navigate to:
   * **Landing Page:** http://127.0.0.1:5000/
   * **User Login:** http://127.0.0.1:5000/login
   * **Admin Login:** http://127.0.0.1:5000/admin (Use admin@example.com and the password you set/the default).

## Project Structure

po/

├── venv/ # Virtual environment (ignored by Git)

├── static/ # Static files (CSS, JS, images)

├── templates/ # HTML templates

│ ├── \_flash\_messages.html

│ ├── admin\_dashboard.html

│ ├── admin\_login.html

│ ├── app\_dashboard.html

│ ├── login.html

│ └── Template1.html # Landing page

├── .env # Environment variables (ignored by Git) - MUST BE CREATED MANUALLY

├── .gitignore # Specifies intentionally untracked files that Git should ignore

├── app.py # Main Flask application file

├── Procfile # For Heroku/Render deployment (defines how to run the app)

└── requirements.txt # Python package dependencies

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487).

IGNORE\_WHEN\_COPYING\_END

## Notes

* The application currently uses an in-memory dictionary for some data if not fully migrated to the database (e.g., USERS\_DB if the DB seeding for users isn't used, and dummy\_database for initial PO data before admin entries, though the aim is to replace these with database operations). The target is for all persistent data to reside in the PostgreSQL database.
* The temp/ folder is created by app.py for temporary file uploads and should ideally be in .gitignore.
* For production deployments (e.g., on Render), environment variables (DATABASE\_URL, SECRET\_KEY, etc.) should be set through the hosting platform's dashboard, not via a .env file.

You can save this as README.md in the root of your po project. When someone clones the repository, this README.md will be the first thing they see on GitHub and will guide them through the setup. Remember to update it if your setup process changes.