TO SET UP THE APP

## Setup

1. **Clone the repository**
2. Create the target directory. In VS code, select the new folder, and in a new terminal window type (with a “.” At the end to avoid creating a directory inside the folder): git clone <https://github.com/yourusername/ai-resume-matcher.git> **.**

The URL for the repo can be copied from the github repo.

To point VS code to my repo (just cloned)

To where VS code is currently pointing: git remote -v

To change where VS code is pointing: git remote set-url origin <https://github.com/YOUR-USERNAME/YOUR-REPOSITORY.git>

To check if VS code is pointing to the right repo: git remote -v

To commit and put to repo:

git add .

git commit -m "Your commit message"

git push -u origin main

1. **Create a virtual environment**
2. python -m venv venv

source venv/bin/activate # On Windows: venv\Scripts\activate

1. **Install dependencies**

pip install -r requirements.txt

1. **Configure environment variables**

Create a .env file in the root directory with the following variables:

FIRECRAWL\_API\_KEY=your\_firecrawl\_key

ANTHROPIC\_API\_KEY=your\_anthropic\_key

DISCORD\_WEBHOOK\_URL=your\_discord\_webhook\_url

RESUME\_URL=your\_resume\_pdf\_url

SUPABASE\_URL=your\_supabase\_url

SUPABASE\_KEY=your\_supabase\_key

CHECK\_INTERVAL\_MINUTES=15

1. **Set up Supabase**

Create a new Supabase project and create the following table:

create table job\_sources (

url text primary key,

last\_checked timestamp with time zone

);

## Running Locally

1. **Start the Streamlit web interface**

streamlit run app.py

1. **Run the job checker scheduler**

python -m src.scheduler

## Deployment

### GitHub Actions Scheduler

The project includes a GitHub Actions workflow that runs the job checker on a schedule. To set it up:

1. Add all environment variables from the .env file as GitHub repository secrets
2. The scheduler will run automatically every Monday (configurable in .github/workflows/scheduler.yml)

### Manual Deployment

1. Set up a server with Python 3.10+
2. Clone the repository and follow the local setup steps
3. Use a process manager like PM2 or Supervisor to run:
4. # For the web interface
5. streamlit run app.py
6. # For the scheduler

python -m src.scheduler

## Project Structure

* app.py: Main Streamlit web application
* src/
  + scraper.py: Job and resume parsing logic
  + matcher.py: AI-powered job matching
  + discord.py: Discord notification system
  + database.py: Supabase database operations
  + models.py: Pydantic data models
  + scheduler.py: Automated job checking