Jobben.de Installation Guide

This guide will walk you through setting up the Jobben.de project both locally for development and on a production server. The installation covers all dependencies, environment configuration, and optional Docker containerization for easy setup.

1. Local Setup

Prerequisites

Before starting, ensure you have the following installed:

- Python 3.8+: Download from python.org.
- MySQL: Ensure you have MySQL running locally or use a remote MySQL instance.

Step-by-Step Instructions

1. Clone the Repository

Start by cloning the Jobben.de project from your version control system (e.g., GitHub).

https://github.com/ErastoMukasa/Jobben.de-project-for-software-development.git

cd jobben-de

2. Create and Activate a Virtual Environment

It's recommended to use a virtual environment to manage dependencies.

```
# Create virtual environment
python3 -m venv venv

# Activate virtual environment (Linux/macOS)
source venv/bin/activate

# Activate virtual environment (Windows)
venv\Scripts\activate
```

3. Install Dependencies

All project dependencies are listed in the requirements.txt file.

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

Dependencies include:

- Flask: Web framework
- SQLAlchemy: ORM for database management
- Flask-Migrate: Database migration tool
- Flask-Login: User session management
- Werkzeug: Security and utility library
- WTForms: Form handling
- Bcrypt: Password hashing
- Flask-Paginate: Pagination of job listings

4. MySQL Database Setup

Create a new MySQL database for the project:

sql

CREATE DATABASE jobben_database;

Ensure the database is up and running with the necessary privileges.

5. Configure Environment Variables

In the project directory, create a .env file to configure environment variables.

touch .env

Add the following content to .env:

FLASK_APP=app.py

FLASK_ENV=development

SECRET_KEY=your_secret_key

SQLALCHEMY_DATABASE_URI=mysql+pymysql://root:@localhost/jobben_database

UPLOAD_FOLDER=uploads

ALLOWED_EXTENSIONS=pdf

The UPLOAD_FOLDER is used for uploading resumes in PDF format.

6. Initialize the Database

Run the following command to create the necessary tables in the MySQL database:

flask db upgrade

This will apply any migrations and ensure the database schema is up to date.

7. Run the Application

Finally, start the Flask development server:

flask run

The app will be accessible at http://localhost:5000/.

2. Production Setup

Prerequisites

In addition to the local setup prerequisites, you'll need:

- A web server (e.g., Nginx, Apache)
- Gunicorn: Python WSGI HTTP Server
- MySQL: Hosted or locally managed
- Supervisor: Process control system (optional, for managing Gunicorn)

Step-by-Step Instructions

1. Install Dependencies

Make sure all dependencies are installed globally or in a virtual environment on the production server:

pip install -r requirements.txt

2. Set Environment Variables

Create the same .env file as used in local development, but adjust for production settings:

FLASK_ENV=production

SECRET_KEY=your_production_secret_key

SQLALCHEMY_DATABASE_URI=mysql+pymysql://root:@localhost/jobben_database

UPLOAD_FOLDER=/var/www/jobben-de/uploads

3. Configure Gunicorn

Install Gunicorn for serving the Flask app in production.

pip install gunicorn

Run the application with Gunicorn:

gunicorn -w 4 -b 0.0.0.0:8000 app:app

- -w 4: Specifies the number of worker processes.
- -b 0.0.0.0:8000: Binds the application to the specified address and port.

4. Setup Nginx

Configure Nginx to reverse proxy requests to Gunicorn. Here’s an example Nginx configuration:

nginx

```
server {
    listen 80;
    server_name Jobben.de;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /static {
        alias /path_to_your_static_directory;
    }
}
```

5. Manage the Application with Supervisor (Optional)

Use Supervisor to manage the Gunicorn process and ensure it stays running:

ini

```
[program:jobben]
command=/path/to/venv/bin/gunicorn -w 4 -b 127.0.0.1:8000 app:app
directory=/path/to/jobben
autostart=true
autorestart=true
```

stderr_logfile=/var/log/gunicorn.err.log

stdout_logfile=/var/log/gunicorn.out.log

6. SSL Configuration (Optional)

For production, it’s highly recommended to enable SSL using Certbot and Nginx:

sudo certbot --nginx -d yourdomain.com

3. Configuration Files

config.py

The config.py file allows you to manage different configurations for development and production.

```
import os

class Config:
    SECRET_KEY = os.getenv('SECRET_KEY')
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    UPLOAD_FOLDER = os.getenv('UPLOAD_FOLDER')
    ALLOWED_EXTENSIONS = {'pdf'}

class DevelopmentConfig(Config):
    DEBUG = True
    SQLALCHEMY_DATABASE_URI = os.getenv('SQLALCHEMY_DATABASE_URI')

class ProductionConfig(Config):
    DEBUG = False
    SQLALCHEMY_DATABASE_URI = os.getenv('SQLALCHEMY_DATABASE_URI')

config = {
    'development': DevelopmentConfig,
    'production': ProductionConfig
}
```

.env

This file stores environment variables like:

SECRET_KEY=your_secret_key

SQLALCHEMY_DATABASE_URI=mysql+pymysql://user:password@localhost/jobben_database

4. Docker Setup (Optional)

Dockerfile

Dockerfile for containerizing the Jobben.de project:

dockerfile

```
FROM python:3.8-slim

# Set the working directory
WORKDIR /app

# Copy the requirements file and install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the project
COPY . .

# Expose port 5000
EXPOSE 5000

# Command to run the app
CMD ["flask", "run", "--host=0.0.0.0"]
```

docker-compose.yml

Use Docker-Compose for multi-container setup:

yaml

```
version: '3'

services:
  web:
    build: .
    ports:
      - "5000:5000"
    environment:
      - FLASK_ENV=development
      - SECRET_KEY=your_secret_key
      - SQLALCHEMY_DATABASE_URI=mysql+pymysql://root:password@db/jobben_database
    depends_on:
      - db

  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: jobben_database
    ports:
      - "3306:3306"
```

Running Docker Containers

Build and run the containers:

```
docker-compose up --build
```