
1. Installation and Configuration Guide: Resume Screening AI

This guide details the prerequisites and core configuration requirements for the Resume AI analyzer system, which operates as a web application utilizing an AI-powered backend.

Prerequisites:

To run or deploy the Resume AI Analyzer application, the following components must be secured and available:

Prerequisite	Description
Python	Version 3.8 or higher is required to run the backend and its dependencies.
OpenAI API Key	Essential for the AI Integration service, as the system calls the OpenAI API for intelligent analysis and scoring. This is necessary for the cloud-hosted Large Language Model (LLM).
Document Parsing Dependencies	Specific Python libraries (like <code>pdfplumber</code> , used by the backend's <code>/upload</code> endpoint) are needed to extract raw text content from uploaded resumes (PDF).

Configuration Procedures

The system's three main architectural components—Web Application, Backend Processing, and AI Integration—require specific configuration.

1. Backend Configuration (Python Flask REST API)

The backend is built using the **Flask framework** and manages file parsing, validation logic, and integration with the AI model.

- **Folder Structure:** The Flask application uses a structured file layout organized by static assets, templates, and services. Key folders include `static`, `templates`, and `uploads`.
- **Templates:** Flask renders HTML templates using Jinja2. Templates include `main.html` (homepage), `loading.html` (intermediate progress page), and `results.html` (final analysis dashboard).
- **Routing:** The backend exposes REST-style API routes, including the `/upload` endpoint to extract text and the `/analyze` endpoint to prompt the OpenAI API.
- **Document Processing:** The Flask backend handles file uploads in PDF and DOCX formats.

2. AI Integration and Environment-Specific Adjustments

The core analysis relies on connecting to an external AI service.

- **API Key Management:** Environment variables, such as API keys and secret keys, must be securely added in the hosting dashboard.
- **AI Model Selection:** The system utilizes the OpenAI API. The default model is the configurable GPT5mini model.
- **Prompt Optimization:** Prompt engineering was performed to improve and sanitize the prompts sent to the API, ensuring accuracy and quality of feedback.

3. Deployment Configuration

The application was designed with deployment in mind.

- **Deployment Setup:** Deployment involved uploading the project files, configuring environment variables, installing dependencies, and launching the application using a production-ready WSGI server that uses `app.py` as the entry point.
- **Access:** After a successful build, a live public URL was generated, making the service accessible to users.

(Note: Screenshots detailing step-by-step local installation could not be provided as the sources only contain architectural diagrams and high-level prerequisites, not detailed setup instructions or images of the local installation process.)
