Professional ATS Resume Scoring System - Documentation

1. Project Overview

This documentation provides comprehensive documentation for the **Professional ATS Resume Scoring System**, a powerful tool built with Microsoft AutoGen and Streamlit. The system is designed to analyze resumes against job descriptions, providing consistent, detailed scoring and actionable feedback for improvement.

The core of the system is a multi-agent architecture powered by AutoGen, where five specialized agents collaborate to deliver a comprehensive analysis. This approach ensures a modular, scalable, and highly efficient workflow.

1.1. System Architecture

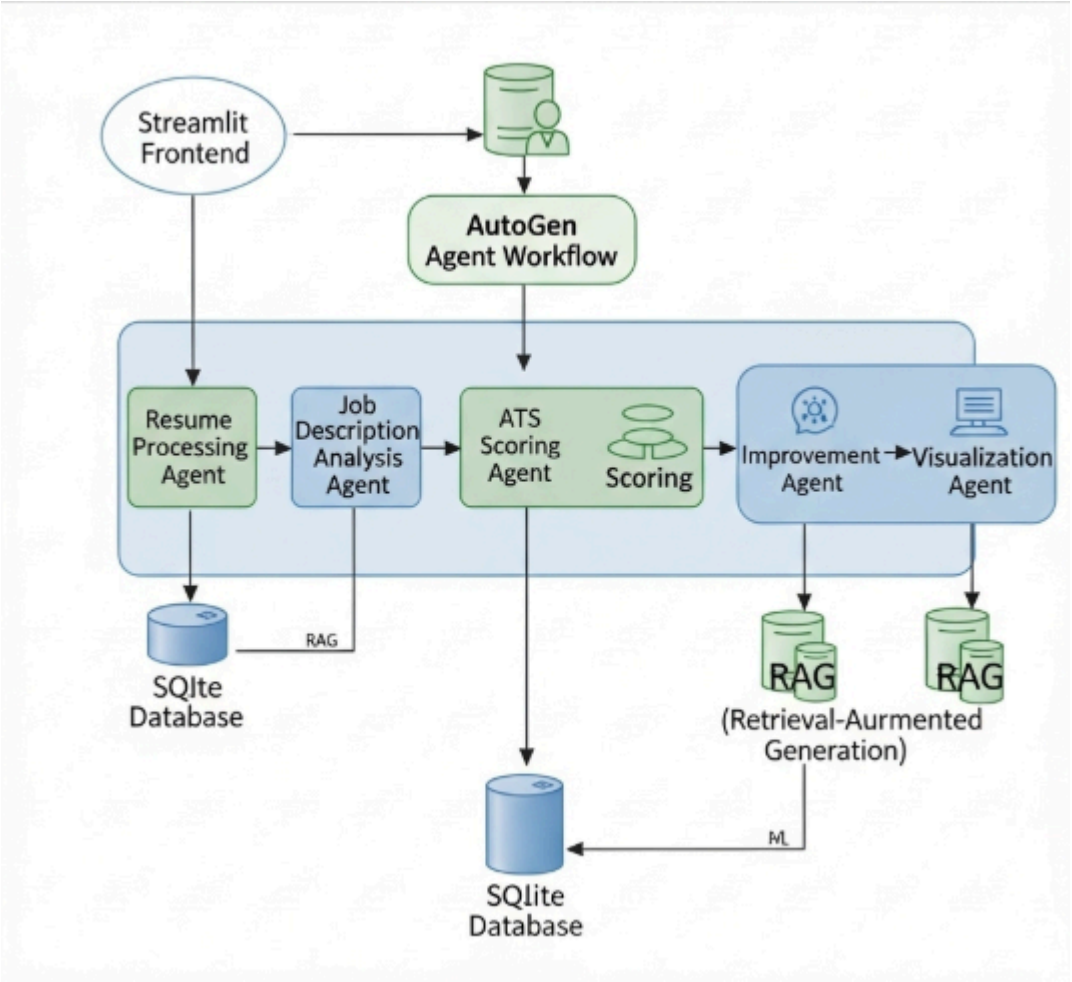
The system is built upon a five-agent architecture:

1. **Resume Processing Agent**: Parses resumes from various formats (PDF, DOCX, TXT) and extracts structured data.
2. **Job Description Analysis Agent**: Analyzes job descriptions to identify key requirements, skills, and keywords.
3. **ATS Scoring Agent**: Implements a consistent, weighted algorithm to score resumes against job requirements.
4. **Improvement Recommendation Agent**: Provides specific, actionable suggestions to improve the resume's score.
5. **Visualization Agent**: Creates professional, easy-to-understand charts and graphs to present the analysis results.

These agents work in concert, orchestrated by the main application logic, and leverage a SQLite database for data persistence and a Retrieval-Augmented Generation (RAG) system for enhanced, context-aware recommendations.

1.2. System Architecture Diagram

The following diagram illustrates the flow of data and interaction between the components of the ATS system.



Flow Description:

1. The user uploads a resume and job description via the Streamlit UI.
2. The `ProfessionalATSSystem` class orchestrates the workflow.
3. The **Resume Processing Agent** and **Job Description Analysis Agent** run in parallel to parse the inputs into structured JSON.
4. This structured data is passed to the **ATS Scoring Agent**, which calculates the scores.
5. The scores, along with the structured data, are then sent to the **Improvement Recommendation Agent**, which leverages the RAG system by querying the database for industry best practices.
6. The final scores and recommendations are used by the **Visualization Agent** to generate charts.
7. All results are compiled and displayed back to the user in the Streamlit interface.

2. Setup and Installation

Follow these steps to set up and run the Professional ATS Resume Scoring System on your local machine.

2.1. Prerequisites

- Python
- Pip (Python package installer)

2.2. Installation

Clone the repository (or download the source code):
git clone <your-repository-url>
cd <your-repository-directory>

1.

Create and activate a virtual environment (recommended):
python -m venv venv
source venv/bin/activate # On Windows, use `venv\Scripts\activate`

2.

Install the required dependencies:
pip install -r requirements.txt

3.

Note: A `requirements.txt` file should be created containing all the imported libraries in `ats_system.py`.

2.3. Configuration

1. **LLM Provider API Key**: The application requires an API key from a supported LLM provider (Groq, DeepSeek, Together AI). This key is entered directly in the Streamlit application's sidebar.
2. **Database**: The system uses a local SQLite database (`resume_ats.db`), which is created automatically when the application is first run. No manual database setup is required.

2.4. Running the Application

Once the dependencies are installed and you have your API key, you can run the Streamlit application with the following command:

streamlit run ats_system.py

The application will open in your default web browser.

3. The 5 AutoGen Agents

This section details the roles and responsibilities of each of the five specialized agents in the system.

3.1. Resume Processing Agent

- **Role**: To act as the initial data processor, converting unstructured resume text into a standardized JSON format.
- **Responsibilities**:
    - Parses resumes in PDF, DOCX, and TXT formats.
    - Extracts key information such as personal details, skills, work experience, and education.
    - Normalizes the extracted text to ensure consistency across different file formats.
    - Outputs a standardized JSON object for downstream processing.
- **Key Feature**: This agent is designed for **consistency**. It uses a zero-temperature setting for the LLM to ensure that the same resume content always produces the identical JSON output, regardless of the original file format.

3.2. Job Description Analysis Agent

- **Role**: To analyze and structure the requirements from a given job description.
- **Responsibilities**:
    - Parses unstructured job description text.
    - Identifies required and preferred skills, experience levels, and educational requirements.
    - Extracts key keywords that are likely to be important for ATS matching.
    - Outputs a standardized JSON object representing the job's requirements.
- **Key Feature**: This agent is crucial for establishing the "ground truth" against which the resume is evaluated. Its consistency ensures a fair and repeatable analysis.

3.3. ATS Scoring Agent

- **Role**: To provide a consistent and objective score for a resume based on the job requirements.
- **Responsibilities**:
    - Implements a weighted scoring algorithm based on the assignment's requirements (Skills: 30%, Experience: 25%, Education: 15%, Format: 15%, Keyword Optimization: 15%).
    - Generates a detailed breakdown of scores for each category.
    - Maintains scoring consistency across multiple runs for the same resume and job description pair.
- **Key Feature**: This agent is the core of the ATS system. Its deterministic nature is enforced by using a zero-temperature LLM and providing clear, mathematical scoring instructions in its system message.

3.4. Improvement Recommendation Agent

- **Role**: To act as a career coach, providing actionable feedback to the user.
- **Responsibilities**:
    - Compares the processed resume data against the job requirements.
    - Identifies missing keywords, skill gaps, and areas for content enhancement.
    - Leverages the RAG system to provide industry-specific advice.
    - Prioritizes recommendations to highlight the most impactful changes.
- **Key Feature**: This agent goes beyond simple scoring by providing constructive feedback, helping users improve their resumes and increase their chances of success.

3.5. Visualization Agent

- **Role**: To translate complex data into intuitive and professional visualizations.
- **Responsibilities**:
    - Creates a radar chart comparing the user's scores against industry benchmarks.
    - Generates a skill match heatmap to visually represent the alignment between the resume and job description.
    - Produces a priority chart for the improvement recommendations.
- **Key Feature**: This agent is not an LLM-based agent but a Python class that uses the Plotly library. It is responsible for the user-facing presentation of the analysis, making the results easy to understand at a glance.

4. User Manual

This section provides a step-by-step guide to using the Professional ATS Resume Scoring System.

1. **Configuration**:
    - Open the application in your browser.
    - In the sidebar, select your preferred LLM provider from the dropdown menu.
    - Enter your API key in the provided text input.
2. **Upload Resume**:
    - Drag and drop your resume file (PDF, DOCX, or TXT) into the file uploader, or click to browse your files.
3. **Enter Job Description**:
    - Copy the full text of the job description you are targeting and paste it into the "Job Description" text area.
4. **Run Analysis**:
    - Click the "Run ATS Analysis" button. The system will now process your resume and the job description through the five-agent workflow. You will see status updates as the analysis progresses.
5. **Review Results**:
    - The results will be displayed in a series of tabs, explaining each feature of the application.

4.1. Troubleshooting Guide

Issue	Possible Cause	Solution
Application fails to start	Dependencies not installed correctly.	Run `pip install -r requirements.txt` in your activated virtual environment.
LLM Connection Error	Invalid or missing API key.	Ensure you have entered a valid API key for the selected provider in the sidebar. Check for typos.
File Upload Fails	Unsupported file format or corrupted file.	Make sure the file is a `.pdf`, `.docx`, or `.txt` file and is not corrupted. Try resaving the file.
Analysis takes a long time	Slow LLM provider response or large file.	Be patient, especially during peak times. The Groq provider is generally the fastest option.
Inconsistent or strange output	LLM provider issue or a bug in the prompt.	Try running the analysis again. If the problem persists, consider restarting the Streamlit application.
Charts not displaying correctly	A potential issue with the Plotly library.	Ensure all dependencies were installed correctly. A page refresh (`F5`) in the browser often resolves rendering issues.

5. Technical Documentation

5.1. API Documentation (Core Classes)

Since this is a Streamlit application and not a web service with a REST API, this section documents the main classes and their public methods.

- **`ProfessionalATSSystem`**
  - `__init__(self, llm_config_list)`: Initializes the database and all five agents.
  - `full_resume_analysis(self, resume_text, job_description, ...)`: The main entry point for the analysis workflow. It orchestrates the calls to the various agents and returns a dictionary with the complete results.
- **`ResumeProcessingAgent`**
  - `process_resume(self, resume_text)`: Takes raw resume text and returns a structured JSON object.
- **`JobAnalysisAgent`**
  - `analyze_job_description(self, job_description)`: Takes raw job description text and returns a structured JSON object.
- **`ATSScoringAgent`**
  - `calculate_score(self, resume_data, job_requirements, ...)`: Takes the structured resume and job data and returns a dictionary of scores.
- **`ImprovementAgent`**
  - `generate_recommendations(self, resume_data, job_requirements, scores)`: Generates and returns a dictionary of actionable improvement suggestions.
- **`ResumeDatabase`**
  - Provides methods like `save_resume`, `save_scores`, `check_existing_score`, etc., for interacting with the SQLite database.

5.2. Database Schema

The application uses a SQLite database to store and manage data. The database is automatically created and populated with sample data on the first run.

- **`resumes`**: Stores uploaded resume information, including the original content, normalized content, and extracted JSON data.
- **`scores`**: Tracks the scoring history for each resume-job description pair, ensuring consistency and allowing for historical analysis.
- **`job_descriptions`**: Stores analyzed job descriptions.
- **`job_templates`**: Acts as the knowledge base for the RAG system, containing industry-specific requirements and best practices.
- **`user_sessions`**: For managing user sessions (a feature for future expansion).
- **`open_jobs`**: A database of sample open jobs used for the job recommendation feature.

5.3. RAG Implementation

The system uses a Retrieval-Augmented Generation (RAG) approach to provide more contextually relevant and insightful recommendations.

- **Knowledge Base**: The `job_templates` table in the SQLite database serves as the knowledge base. It is pre-populated with sample data containing industry-specific keywords, requirements, and best practices.
- **Retrieval**: When the **Improvement Recommendation Agent** is activated, it retrieves relevant information from the knowledge base based on the industry identified in the resume.
- **Generation**: This retrieved information is then passed to the LLM as part of the prompt, allowing the agent to generate recommendations that are not only based on the immediate resume and job description but also on broader industry standards.

5.4. Scoring Algorithm

The scoring is designed to be **consistent and deterministic**. The weights for each category are hard-coded in the **ATS Scoring Agent's** system message to ensure they are applied consistently:

- **Skills Match**: 30%
- **Experience Relevance**: 25%
- **Education Alignment**: 15%
- **Format and Structure**: 15%
- **Keyword Optimization**: 15%

The agent is instructed to use mathematical calculations rather than subjective judgment, and the LLM's temperature is set to 0.0 to minimize randomness and ensure that identical inputs produce identical outputs.

6. Testing and Performance

6.1. Consistency Testing

- **Content Hashing**: The application generates MD5 hashes of the normalized resume and job description content.
- **Caching**: Before calculating a new score, the system checks the `scores` table for an existing entry with the same content hashes. If a match is found, the cached score is returned, guaranteeing consistency and improving performance.
- **Deterministic Prompts**: The prompts for the scoring and processing agents are designed to be deterministic, with a zero-temperature setting for the LLM.

6.2. Demo and Testing Results

The following table shows the results from the live demonstration, using five sample resumes against a standardized "Software Engineer" job description.

Resume File Name	Final Score (/100)
resume_sample_1	**74.2**
resume_sample_2	**54.5**
resume_sample_3	**72.2**
resume_sample_4	**64.5**
resume_sample_5	**54.5**

6.3. Performance Requirements

The system is designed to meet the following performance targets as specified in the assignment:

- **Resume processing**: < 30 seconds
- **Score calculation**: < 10 seconds
- **UI responsiveness**: The Streamlit front-end is designed to be lightweight and responsive.

7. Future Enhancements

The following are potential bonus features that could be implemented to extend the system's capabilities:

- **AI-Powered Resume Rewriting**: Suggest specific text improvements and rephrasing.
- **Batch Processing**: Allow users to upload and analyze multiple resumes at once.
- **API Integration**: Connect to job boards like LinkedIn or Indeed to fetch job descriptions in real-time.
- **Advanced Analytics**: Provide users with trend analysis of their scores over time and across different job applications.