Software Requirements Specification (SRS)

for AI Resume Analyzer (Full-Stack + ML)

Version: 1.0

Date: January 12, 2026 Status: Draft

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the **AI Resume Analyzer**. This system is designed to automate the initial screening process of recruitment by parsing resumes, extracting key information, and scoring candidates against specific job descriptions using Machine Learning (ML) and Natural Language Processing (NLP).

1.2 Scope

The AI Resume Analyzer is a web-based application that serves two primary user groups: Candidates and Recruiters.

- **For Candidates:** The system analyzes their resume against a target job description, identifying missing keywords and providing suggestions for improvement.
- For Recruiters: The system allows batch uploading of resumes, automatic parsing, and ranking of candidates based on semantic relevance to a job description.
  The system will utilize a Python-based backend for ML processing and a responsive web frontend for user interaction.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirements Specification
- **JD:** Job Description
- **NLP:** Natural Language Processing
- **NER:** Named Entity Recognition
- **ATS:** Applicant Tracking System
- **RBAC:** Role-Based Access Control
- **PII:** Personally Identifiable Information

2. Overall Description

2.1 Product Functions

The major functions of the system include:

- User account management and role separation.
- Resume file uploading and text parsing.
- Job Description management.
- Semantic matching analysis between Resume and JD.
- Gap analysis (identifying missing skills).
- Report generation (PDF export).

2.2 User Classes and Characteristics

- **Candidate:** Tech-savvy or general users looking to improve their resume. They require a simple, drag-and-drop interface and clear, actionable feedback.
- **Recruiter:** HR professionals or hiring managers. They require efficiency, batch processing capabilities, and the ability to sort/filter large lists of candidates.
- **Admin:** System maintainers responsible for user management and system configuration.

2.3 Operating Environment

- **Client:** Modern web browsers (Chrome, Firefox, Safari, Edge) on Desktop and Mobile devices.
- **Server:** Cloud-based infrastructure (AWS/GCP).
- **Database:** PostgreSQL for relational data; Object Storage (S3) for documents.

2.4 Assumptions and Dependencies

- Users have internet connectivity.
- Uploaded resumes are in English (initial release).
- The system depends on external NLP libraries (e.g., Spacy, Transformers) remaining available and compatible.

3. Specific Requirements

3.1 External Interface Requirements

- **User Interface:** The UI shall be implemented using a modern JavaScript framework (React.js/Next.js) and must be responsive (mobile-friendly).
- **Hardware Interface:** No specific hardware interface required beyond standard server infrastructure.
- **Software Interface:**
    - **OS:** Linux-based servers for backend deployment.
    - **API:** RESTful API for communication between frontend and backend.

3.2 Functional Requirements (FR)

3.2.1 User Authentication & Profile

- **FR-01:** The system shall allow users to register and login using Email/Password.
- **FR-02:** The system shall support OAuth login via Google and LinkedIn.
- **FR-03:** The system shall enforce Role-Based Access Control (RBAC) for Candidates, Recruiters, and Admins.
- **FR-04:** The system shall provide a dashboard displaying recent uploads and analysis history.

3.2.2 Resume Upload & Parsing

- **FR-05:** The system shall accept file uploads in .pdf, .docx, and .txt formats.
- **FR-06:** The system shall validate file size (max 5MB) and reject corrupted files.
- **FR-07:** The system shall extract raw text from uploaded files while preserving logical section order.
- **FR-08 (NER):** The system shall automatically identify and categorize:
    - Contact Info (Name, Email, Phone, Links)
    - Education (Degree, University, Year)
    - Skills (Technical, Soft)
    - Experience (Company, Title, Dates)

3.2.3 AI Analysis & Scoring

- **FR-09:** The system shall allow users to input a Job Description (JD) via text paste or file upload.
- **FR-10:** The system shall calculate a Semantic Similarity Score (0-100%) between the resume and JD using NLP models (e.g., BERT).
- **FR-11:** The system shall perform a Keyword Gap Analysis to identify "Hard Matched" skills and "Missing Critical" skills.
- **FR-12:** The system shall provide feedback on section quality (e.g., flagging long summaries, lack of metrics, absence of action verbs).

3.2.4 Reporting & Visualization

- **FR-13:** The system shall display an interactive scorecard visualizing match percentages across different categories (Skills, Experience, Education).
- **FR-14:** The system shall allow users to export the full analysis report as a PDF.
- **FR-15:** For Recruiters, the system shall generate a ranked leaderboard of candidates based on match scores.

3.3 Non-Functional Requirements (NFR)

3.3.1 Performance

- **NFR-01:** The system shall complete the parsing and analysis pipeline within 3-5 seconds per document under normal load.
- **NFR-02:** The system shall support concurrent uploads from at least 50 users simultaneously without degradation.

3.3.2 Reliability & Accuracy

- **NFR-03:** The text extraction module shall achieve >90% accuracy on standard layouts.
- **NFR-04:** The scoring algorithm shall be bias-aware, ignoring demographic data (name, gender, address) during the scoring process.
- **NFR-05:** The system shall maintain 99.9% uptime during business hours.

3.3.3 Security

- **NFR-06:** All sensitive data (resumes, user profiles) shall be encrypted at rest (AES-256).
- **NFR-07:** All data transmission shall occur over HTTPS (TLS 1.2+).
- **NFR-08:** The system shall provide a "Blind Hiring" mode to mask PII for recruiters.

3.3.4 Usability

- **NFR-09:** The application shall be responsive and fully functional on devices ranging from mobile phones to large desktop monitors.
- **NFR-10:** The system shall provide clear, user-friendly error messages for invalid inputs or system failures.

4. Technical Constraints

- **Backend:** Must be built using Python (FastAPI/Django) to leverage NLP libraries.
- **Asynchronous Processing:** Long-running ML tasks must be offloaded to a task queue (Redis + Celery) to prevent request timeouts.
- **Data Retention:** Guest uploads must be automatically purged after 24 hours.

5. Appendices

- **Appendix A:** Database Schema Diagram (To be added)
- **Appendix B:** API Endpoint Documentation (To be added)