Software Requirements Specification (SRS)

Project Title: Applicant Tracking System (ATS)

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for a web-based Job and Candidate Management System for NewRise Technosys. The system will streamline job postings, candidate applications, automated resume shortlisting using an AI-powered Applicant Tracking System (ATS), and maintain candidate interaction and hiring history.

1.2 Scope

The system will:

- Allow HR teams to post jobs and manage them.
- Allow candidates to apply for jobs via a public-facing interface.
- Include an AI-based resume screening and shortlisting system.
- Maintain candidate history including feedback, interviews, and job applications.
- Enable communication and tracking between HR and candidates.
- Include role-based access for Admin, HR, Interviewers, and Candidates.

2. Overall Description

2.1 Product Perspective

The system will be a cloud-hosted, browser-accessible application built with a modern JavaScript frontend framework (e.g., React.js/Next.js) and a scalable backend (e.g., Nest.js + PostgreSQL). It will optionally integrate with external job boards and internal HRMS.

2.2 User Classes and Characteristics

User Role	Description
Admin	Full system access. Manages users, permissions, and reports.
HR Manager	Creates job posts, reviews candidates, manages shortlisting, schedules interviews.
Interviewer	Reviews candidate info, submits interview feedback.
Candidate	Applies for jobs, tracks application status.

3. Functional Requirements

3.1 Job Management

- FR1.1: HR/Admin can create, edit, and archive job posts.
- FR1.2: Each job can have custom fields (skills, experience, salary, etc.).
- FR1.3: Jobs can be tagged and filtered by departments, locations, or tags.

3.2 Candidate Application Portal

- FR2.1: Candidates can view and apply to open jobs.
- FR2.2: Candidate registration with resume and profile details.
- FR2.3: Allow uploading documents (CV, cover letter, etc.).

3.3 AI-Powered ATS

FR3.1: The system shall parse both job descriptions and resumes into structured text formats using a resume parsing library or service.

FR3.2: An AI microservice (Python-based) shall compute a **Resume Match Score** by:

- Extracting keywords from the job description (skills, tools, qualifications, etc.).
- Extracting keywords and entities from resumes.
- Calculating similarity scores based on keyword overlap, semantic similarity (e.g., cosine similarity using TF-IDF or sentence embeddings).

FR3.3: The score shall be presented to HR as a numeric value (e.g., 0–100%) alongside a breakdown of matched and missing keywords.

FR3.4: The AI microservice must provide an API endpoint (`/score-resume`) that accepts resume text and job description, and returns a structured score object:

```
{
  "score": 82,
  "matched_keywords": ["Python", "REST API", "PostgreSQL"],
  "missing_keywords": ["CI/CD", "AWS", "Microservices"]
}
```

FR3.5: HR can sort, filter, and search candidates based on their match scores and keywords.

3.4 Interview & Communication Management

- FR4.1: HR can schedule interviews and assign interviewers.
- FR4.2: Interviewers can record feedback (qualitative and quantitative).
- FR4.3: Email and notification system for candidates and team members.

3.5 Candidate History Management

- FR5.1: Maintain a timeline of every candidate's applications, interviews, outcomes.
- FR5.2: View all feedback, resumes, and job roles applied to in one place.

3.6 Reporting and Analytics

- FR6.1: Dashboard for job openings, candidate status, hiring funnel.
- FR6.2: Exportable reports on hiring metrics, source analysis, time-to-hire, etc.

4. Non-Functional Requirements

4.1 Performance

- The system should respond to user inputs within 2 seconds under normal load.

4.2 Scalability

- The system must handle growth in the number of users, resumes, and job posts without degradation in performance.

4.3 Security

- Implement RBAC (Role-Based Access Control).
- Data encryption in transit (HTTPS) and at rest.
- GDPR-compliant data handling.

4.4 Availability

- The system should maintain 99.9% uptime with automated backups.

4.5 Usability

- Intuitive, mobile-responsive UI.
- User onboarding with tooltips and documentation.

5. External Interfaces

5.1 Web Browser

- Compatible with modern browsers (Chrome, Firefox, Edge, Safari).

5.2 Email Service

- Integration for email notifications (e.g., SendGrid, SMTP).

5.3 Resume Parsing API

- Integration with services like Sovren or Rchilli for structured resume data.

5.4 AI Screening Engine

- Custom or integrated AI model to score resumes based on job description parsing and NLP.

6. System Architecture Overview

- **FastAPI** or **Flask** (Python): For the AI microservice API.
- **spaCy + PyMuPDF** or **pdfminer.six**: Resume parsing if not using a 3rd-party API.
- **PostgreSQL**: Store scores and keywords for analytics and history.
- **RabbitMQ/Kafka (optional)**: For async resume scoring.

- **Frontend**: React.js or Next.js with TypeScript and Tailwind CSS for responsive design.
- **Backend**: Nest.js (Node.js), REST/GraphQL API
- **Auth**: JWT-based authentication + OAuth support (LinkedIn, Google)
- **AI ATS Microservice**:
  - **Language**: Python
  - **Libraries**: `spaCy`, `scikit-learn`, `NLTK`, `transformers` (for semantic embeddings)
  - **Functionality**:
    - Parse and clean resume/job texts.
    - Extract entities and keywords.
    - Calculate keyword match percentage using:
      - TF-IDF Vectorization + Cosine Similarity (Baseline)
      - Optionally, semantic similarity using Sentence Transformers (for future improvements)
  - **Deployment**: As a RESTful microservice using FastAPI or Flask.
  - **Input/Output**:
    - Input: Raw text of resume and job description.
    - Output: JSON object with match score and keyword insights.
  - **Authentication**: Token-based internal API access.
  - **Scalability**: Can be containerized (Docker) and scaled separately.

7. Constraints

- Must be deployed within 1 months.
- Must support localization for at least English and one regional language.
- Hosted on secure cloud infrastructure (AWS, GCP, etc.).

8. Future Enhancements

- Integration with LinkedIn Jobs and Naukri APIs.
- AI chatbot for candidate FAQs.
- Skill assessment test integration.
- Employee referral system module.

9. Glossary

Term	Definition
ATS	Applicant Tracking System, software to manage recruitment processes.
Resume Match Score	A numeric score indicating how well a resume matches a job description.
RBAC	Role-Based Access Control, a method of regulating access to system resources based on user roles.
GDPR	General Data Protection Regulation, EU regulation on data protection and privacy.
API	Application Programming Interface, a set of rules for building software applications.
NLP	Natural Language Processing, a field of AI that focuses on the interaction between computers and human language.
UI	User Interface, the means by which a user interacts with a computer or software.