Project Charter

Applicant Tracking System (ATS) - Resume Analyzer

Prepared for: Project Stakeholders / Academic Supervisor

Prepared by: Data Science/Software Development Capstone Team ([Student's Name(s)])

Project Mentor: [Instructor's Name / Your Name as Mentor]

Version: 1.0

Date: May 16, 2025

Abstract

This document outlines the charter for the "Applicant Tracking System (ATS) - Resume Analyzer" project undertaken by the Capstone Team for FACE Prep Campus. The primary objective is to develop a web application that allows users to upload a resume and a job description. The system will extract text from these documents, calculate a similarity score, identify and display key skills from both, and provide an overall match assessment. The project aims to provide a practical tool for initial resume screening. The core version will leverage Python-based NLP/ML libraries like spaCy and Scikit-learn, with MongoDB for data storage and Streamlit for the user interface. An optional advanced phase may explore the integration of Large Language Models (LLMs) like OpenAI's GPT models (via API and potentially Langchain) for enhanced semantic understanding and feature extraction.

Contents

1 Introduction and Background

Recruiting top talent efficiently is a critical challenge for organizations. Applicant Tracking Systems (ATS) play a vital role in managing the hiring pipeline. A key function of an ATS is to help recruiters quickly assess the suitability of a candidate's resume against a specific job description (JD). This project aims to build a focused application that automates parts of this initial screening process by extracting relevant information, identifying skills, and quantifying the match between a resume and a JD. This tool can help save recruiter time and improve the consistency of initial evaluations.

2 Problem Statement

Manually reviewing a large volume of resumes for multiple job openings is time-consuming and prone to human bias or oversight. Recruiters need a tool that can quickly:

- Extract text from various resume formats (PDF, potentially DOCX).
- Identify key skills present in both the job description and the resume.
- Quantify the similarity or match between the resume and the job description.
- Provide a clear, user-friendly interface for these functionalities.

Existing solutions can be expensive or part of larger, more complex ATS platforms. This project seeks to develop a lean, effective tool focused on these core resume analysis tasks.

3 Project Goals and Objectives

The overarching goal of this project is to develop a functional ATS Resume Analyzer web application that assists in the initial screening of resumes against job descriptions.

3.1 Primary Objectives (Core System)

1. **Document Upload and Text Extraction:**
   - Allow users to upload a resume file (primarily PDF) and a job description file (PDF or text input).
   - Implement robust text extraction from these documents using libraries like PyMuPDF (for text-based PDFs) and potentially Pytesseract (for image-based PDFs).
   - Display the extracted text within the application.
2. **Skill Identification:**
   - Utilize spaCy (or similar NLP libraries) to identify and extract key technical and soft skills from the job description text.
   - Identify and extract skills from the resume text.
   - Display the top N (e.g., 5-10) skills found in the JD and the resume, highlighting matched skills.
3. **Similarity Scoring:**
   - Implement a similarity scoring mechanism (e.g., TF-IDF with Cosine Similarity using Scikit-learn, or sentence embeddings) to quantify the match between the resume and the job description.

- Display the calculated similarity score.

4. **Overall Match Assessment (ML-based):**
   - Develop a simple machine learning model or a weighted scoring heuristic that combines factors like text similarity, number of matched skills, and potentially other derived features to provide an overall match score or category (e.g., "Good Match," "Partial Match," "Low Match").
5. **User Interface:**
   - Develop an intuitive web-based user interface using Streamlit.
6. **Data Persistence (Basic):**
   - Utilize MongoDB (via MongoDB Atlas) to store extracted text, identified skills, and analysis results for uploaded documents (e.g., per session or user if basic user management is considered).

3.2 Secondary Objectives (Potential Extensions / LLM Integration Phase)

1. **Enhanced Skill Extraction with LLMs:**
   - Integrate with an LLM (e.g., OpenAI GPT-3.5/4 via API, potentially using Langchain) to perform more nuanced and context-aware skill extraction from resumes and JDs.
2. **Advanced Semantic Matching with LLMs:**
   - Utilize LLM embeddings or direct LLM judgment (via prompting) for a more sophisticated semantic similarity score or qualitative match assessment.
3. **Automated Feedback/Summary (LLM-based):**
   - Explore using an LLM to generate a brief summary of why a resume is a good/poor fit, or to identify key missing skills.
4. **Support for Additional File Formats:** Extend resume parsing to include .docx files.
5. **User Accounts and History (if time permits):** Basic user authentication and storage of past analyses.

4 Scope of Work

4.1 In-Scope Activities (Core System)

- Design and development of the Streamlit web application.
- Implementation of file upload functionality for resumes (PDF) and job descriptions (PDF/text).
- Development of text extraction modules using PyMuPDF/Pytesseract.
- Development of skill extraction logic using spaCy (NER, Matcher).
- Implementation of text similarity algorithms (e.g., TF-IDF, Cosine Similarity).
- Development of a basic ML model or heuristic for overall resume scoring.
- Setup and integration of MongoDB Atlas for data storage.

- Unit testing of core components.
- Documentation of code and system architecture.
- Final project report and presentation.

4.2 In-Scope Activities (LLM Integration Phase - if pursued)

- Setting up an OpenAI API account and managing API keys securely.
- Learning and implementing Langchain for LLM interaction (if chosen).
- Developing prompts for skill extraction and semantic matching using the LLM.
- Integrating LLM API calls into the application workflow.
- Comparing LLM-based results with core system results.

4.3 Out-of-Scope Activities (Unless explicitly added as extensions)

- Full-fledged commercial-grade ATS features (e.g., candidate pipeline management, interview scheduling, complex user roles and permissions). Automated job board scraping.
- Training of large, custom deep learning models from scratch for NLP tasks (will leverage pre-trained models from spaCy or LLM APIs).
- Advanced UI/UX design beyond Streamlit's standard capabilities without custom components.
- Large-scale performance testing and optimization for thousands of concurrent users.
- Direct integration with cloud storage like AWS S3/Azure Blob for uploaded files (files will be processed in-session or temporarily stored, with metadata/results in MongoDB, unless this becomes a specific requirement).

5 Methodology

The project will follow an agile-inspired, iterative development approach:

1. **Project Initiation & Planning:** Define scope, objectives, tech stack, and deliverables (this document).
2. **Environment Setup:** Set up development environment using `uv` for virtual environment creation (‘uv venv‘) and package installation (‘uv pip install‘). Initialize version control (Git) and set up the MongoDB Atlas instance.
3. **Phase 1: Core System Development**
    a. *Module 1: Document Parsing & Text Extraction:* Implement reliable text extraction from PDFs.
    b. *Module 2: UI Foundation & File Upload:* Basic Streamlit app with upload functionality.
    c. *Module 3: Skill Extraction (spaCy):* Develop logic to identify skills from extracted text.
    d. *Module 4: Similarity Scoring (Traditional):* Implement TF-IDF and Cosine Similarity.

e. *Module 5: Basic Overall Scoring Logic.*

f. *Module 6: MongoDB Integration:* Store and retrieve analysis data.

g. *Module 7: Integration & Testing:* Combine modules and test core functionality.

4. **Phase 2: LLM Integration Advanced Features (Optional/Time-Permitting)**

   a. *Module 8: LLM API Setup Basic Interaction (Langchain/OpenAI SDK).*

   b. *Module 9: LLM-enhanced Skill Extraction.*

   c. *Module 10: LLM-based Semantic Matching/Scoring.*

   d. *Module 11: Integration and Comparative Analysis.*

5. **Final Testing & Refinement:** Thorough testing, bug fixing, and UI improvements.

6. **Documentation & Reporting:** Finalize all documentation.

7. **Project Closure & Handover:** Deliver all project artifacts and presentation.

6 Key Deliverables

- **Functional Web Application:** A Streamlit application deployed locally (or on a simple hosting platform if feasible) demonstrating all implemented features.
- **Source Code Repository:** Well-documented Python code, including Streamlit app files, NLP/ML scripts, and any backend logic, managed via Git (e.g., on GitHub/GitLab).
- **Jupyter Notebooks (for exploration):** Notebooks used for experimenting with PDF parsing, NLP techniques, and model development.
- **Database Schema (MongoDB):** Description of the data structures used in MongoDB.
- **Final Project Report:** A comprehensive document detailing the project's objectives, methodology, architecture, implementation details, results, challenges, and future work. This charter will serve as the initial version.
- **Presentation Slides:** For the final project presentation.
- **User Guide (Basic):** Instructions on how to use the application.

7 Success Metrics / Key Performance Indicators (KPIs)

- **Functionality:** Successful implementation of all core objectives outlined in Section 3.
- **Accuracy of Text Extraction:** Qualitative assessment of how well text is extracted from sample PDFs.
- **Relevance of Skill Extraction:** Qualitative assessment of the relevance and accuracy of skills extracted by spaCy (and LLM, if implemented).
- **Meaningfulness of Similarity Score:** The similarity score should intuitively reflect the match quality for sample resume-JD pairs.
- **Usability:** The Streamlit application should be intuitive and easy to use.
- **Code Quality:** Adherence to PEP 8 guidelines, good commenting, and modular design.

- **Completion of Deliverables:** All items listed in Section 6 are completed and submitted.
- **Adherence to Project Plan:** Timely completion of milestones (within reasonable limits for a capstone).

8 Tools, Technologies, and Resources

8.1 Core Technologies

- **Programming Language:** Python 3.x
- **Web Framework (Frontend):** Streamlit
- **PDF Processing:** PyMuPDF (fitz), Pytesseract (with Tesseract OCR), pdf2image (for Pytesseract)
- **NLP Library:** spaCy (for tokenization, NER, rule-based matching)
- **Machine Learning Library:** Scikit-learn (for TF-IDF, Cosine Similarity, other ML models)
- **Database:** MongoDB (via MongoDB Atlas cloud service)
- **MongoDB Python Driver:** Pymongo

8.2 Potential LLM-Phase Technologies

- **LLM API Access:** OpenAI API (for GPT-3.5/GPT-4 models)
- **LLM Orchestration (Optional):** Langchain
- **HTTP Requests Library (for API calls):** ‘requests‘ (if not using Langchain’s built-in clients)

8.3 Development Environment Tools

- **IDE:** VS Code, PyCharm, or Jupyter Lab/Notebooks
- **Version Control:** Git, GitHub/GitLab
- **Python Environment & Package Management:** `uv` (for creating virtual environments with ‘uv venv‘ and installing packages with ‘uv pip install‘). Alternatively, traditional ‘venv‘/‘pip‘ or ‘conda‘.
- **Operating Systems (Development/Deployment):** Windows, Linux, macOS

8.4 Dataset (for testing and initial skill lists)

- Sample resumes (publicly available examples, or anonymized ones if ethical).
- Sample job descriptions from various roles/industries.
- Potentially pre-defined lists of common skills for bootstrapping the skill extractor.

9 Project Team Roles (Illustrative)

(Adjust based on team size; for a single student, they'd cover all technical roles)

- **Project Lead / Full-Stack Developer:** [Student's Name(s)] - Responsible for overall project management, frontend (Streamlit), backend logic, NLP/ML model implementation, and database integration.
- **NLP/ML Specialist (if team > 1):** Focus on text extraction, skill identification algorithms, similarity scoring, and ML model development.
- **Database/Backend Developer (if team > 1):** Focus on MongoDB schema design, data persistence logic, and API integrations (if any beyond OpenAI/MongoDB).
- **Project Sponsor/Mentor:** [Instructor's Name] - Provides guidance, feedback, and academic oversight.

10 Timeline Milestones (Illustrative - e.g., for a 12-16 week project)

- **Week 1-2:** Project Kick-off, Detailed Planning, Requirements Refinement, Tech Stack Setup, Initial research on PDF parsing and NLP libraries.
- **Week 3-4:** Basic Streamlit App, File Upload, PDF Text Extraction (Core).
- **Week 5-6:** Skill Extraction using spaCy (from JD and Resume).
- **Week 7-8:** TF-IDF & Cosine Similarity Implementation, Display Scores.
- **Week 9-10:** Basic Overall Scoring Model, MongoDB Integration for results.
- **Week 11-12 (Optional - Start LLM Phase or Refinement):**
  - If LLM Phase: OpenAI API setup, basic LLM calls for skill extraction/similarity.
  - If No LLM Phase: Refine core features, robust testing, UI improvements.
- **Week 13-14 (Optional - Continue LLM or Finalize):**
  - If LLM Phase: Integrate LLM features, comparative analysis.
  - If No LLM Phase: Final testing, documentation.
- **Week 15-16:** Finalize Report, Prepare Presentation, Project Submission.

(A more detailed Gantt chart or project plan may be appended separately)

11 Reporting Communication Plan

- **Weekly/Bi-Weekly Progress Updates:** To Project Mentor via email or brief meetings.
- **Mid-Project Review/Demo:** Presentation of progress on core functionalities.
- **Final Project Presentation & Demo:** Formal presentation to stakeholders/supervisor.

12 Ethical Considerations

- **Data Privacy & Security (Resumes):**
  - If using real resumes for development/testing, they must be anonymized or consent obtained.
  - The application should clearly state how uploaded data is handled (e.g., processed in-session, stored temporarily, or if persisted in MongoDB, how it's secured).
  - No personally identifiable information (PII) beyond what's necessary for analysis should be stored long-term without explicit user consent and security measures.
  - Secure handling of API keys (OpenAI, MongoDB Atlas connection strings).
- **Bias in Algorithms:**
  - NLP models (including spaCy pre-trained models and LLMs) can inherit biases from their training data. Be aware that skill extraction or scoring might inadvertently favor certain phrasings or backgrounds.
  - The project should acknowledge these potential biases and avoid making definitive hiring recommendations. The tool is an aid, not a replacement for human judgment.
- **Transparency:**
  - Clearly explain to the user what the application does (e.g., how similarity is calculated, what skills are being looked for).
  - Acknowledge limitations of the automated analysis.
- **Responsible AI Use (especially if using LLMs):** Ensure LLM outputs are reviewed for appropriateness and don't generate misleading or harmful content. Avoid over-reliance on automated scores.

13 Learning Outcomes for Capstone Team

- Practical experience in developing a full-stack web application using Python and Streamlit.
- Proficiency in document parsing techniques for various file formats (PDF).
- Applied knowledge of Natural Language Processing (NLP) techniques using spaCy for tasks like text processing, named entity recognition (skill extraction), and rule-based matching.
- Understanding and implementation of text similarity measures (e.g., TF-IDF, Cosine Similarity, potentially embedding-based similarity).
- Experience with NoSQL databases (MongoDB Atlas) for storing and retrieving application data.
- (If LLM phase is included) Experience with integrating and utilizing Large Language Model APIs (e.g., OpenAI) and orchestration tools (e.g., Langchain) for advanced NLP tasks.
- Skills in project planning, agile development practices, version control (Git), and technical documentation.

- Enhanced ability to identify, analyze, and solve real-world problems using data science and software engineering principles.
- Understanding of ethical considerations in developing AI/ML applications, particularly with sensitive data like resumes.

Approvals

[Student's Name(s)]	[Instructor's Name / Mentor Name]
Data Science/Software Capstone Team	Project Lead / Mentor
FACE Prep Campus	FACE Prep Campus
Date: _____	Date: _____