Smart Recruitment Assistant: An AI-Driven Recruitment Platform Combining Semantic Matching and Retrieval-Augmented Generation

JAAFAR Wafa, ATIGUI Yassine, YADANI Adnane.

Excellence Master's Program in Data Analytics and Artificial Intelligence (ADIA), Ibn Zohr University
GitHub Repository: `https://github.com/YsneAtigui/Smart-Recruitment-Assistant/`

Abstract :

The recruitment process increasingly relies on the analysis of large volumes of unstructured data, primarily in the form of resumes and job descriptions. Traditional Applicant Tracking Systems (ATS) mainly depend on keyword-based matching techniques, which often fail to capture the semantic meaning of candidate profiles and job requirements, leading to suboptimal hiring decisions.

This academic project, conducted within the *Data Science* module, presents **Smart Recruitment Assistant**, an AI-driven recruitment platform designed to address these limitations by combining semantic analysis, multi-dimensional matching, and Retrieval-Augmented Generation (RAG). The proposed system automatically extracts and structures information from resumes and job descriptions, generates semantic embeddings, and computes an interpretable matching score based on multiple criteria, including skills, experience, education, and overall semantic similarity.

Furthermore, the platform integrates a RAG-based question-answering mechanism that enables recruiters and candidates to interact with recruitment data using natural language queries, providing context-aware and explainable responses supported by explicit source attribution. The system is implemented using a full-stack architecture combining React, FastAPI, SQLite, and ChromaDB.

The results demonstrate that the proposed approach improves the accuracy, transparency, and usability of recruitment analysis, offering a robust and extensible foundation for intelligent decision support in modern hiring workflows.

1 Introduction

1.1 Background and Motivation

The rapid digitalization of recruitment processes has led to the accumulation of vast amounts of unstructured textual data, primarily in the form of resumes and job descriptions. Recruiters are often required to manually analyze and compare dozens, or even hundreds, of candidate profiles for a single job opening. This task is time-consuming, error-prone, and susceptible to subjective bias.

To address these challenges, many organizations rely on Applicant Tracking Systems (ATS). However, most existing ATS solutions are based on simple keyword matching and rule-based filtering techniques. Such approaches are limited in their ability to understand the semantic meaning of text, often overlooking relevant candidates whose resumes do not explicitly contain predefined keywords.

1.2 Problem Statement

The core limitation of traditional recruitment systems lies in their inability to perform deep semantic reasoning over unstructured documents. Keyword-based matching fails to account for variations in terminology, contextual relevance, and implicit skills or experiences. As a result, candidates with strong profiles may be incorrectly filtered out, while less suitable candidates may be ranked highly.

Moreover, conventional systems provide little to no explainability regarding matching decisions. Recruiters receive a score or ranking without insight into why a candidate was selected or rejected. Similarly, candidates lack actionable feedback on how their profiles align with specific job requirements.

1.3 Objectives of the Project

This academic project, developed as part of the *Data Science* module, aims to design and implement an intelligent recruitment assistant that addresses the aforementioned limitations. The primary objectives of the project are as follows:

- Automatically extract and structure information from resumes and job descriptions using natural language processing techniques.
- Compute an interpretable and multi-dimensional matching score between candidates and job offers.
- Enable natural language interaction with recruitment data through a Retrieval-Augmented Generation (RAG) framework.
- Support both recruiters and candidates with tailored insights, analytics, and recommendations.

1.4 Contributions

The main contributions of this project can be summarized as follows:

- An end-to-end AI-driven recruitment platform integrating document parsing, semantic representation, and intelligent matching.
- A multi-dimensional CV–Job matching approach combining semantic similarity, skills alignment, experience, and education.
- A RAG-based question-answering system enabling explainable and context-aware interaction.
- A hybrid data architecture combining a relational database (SQLite) and a vector database (ChromaDB).
- A dual-persona system supporting both recruiters and job seekers.

2 Related Work

Automated recruitment systems have traditionally relied on Applicant Tracking Systems (ATS) that perform keyword-based filtering and rule-based ranking of resumes. While these systems improve processing efficiency, they often lack semantic understanding and may overlook qualified candidates due to vocabulary mismatch.

Recent advances in Natural Language Processing have enabled the use of semantic embeddings to represent resumes and job descriptions in a continuous vector space. These approaches improve matching accuracy by capturing contextual meaning rather than exact word overlap. However, most embedding-based systems provide limited interpretability and do not support interactive exploration of candidate profiles.

More recent research explores the use of Large Language Models to extract structured information from unstructured recruitment data and generate summaries or recommendations. While these models offer strong reasoning capabilities, they may produce hallucinated outputs when used without grounding.

Retrieval-Augmented Generation has emerged as a promising solution to address this limitation by combining semantic retrieval with generative models. Although RAG has been widely applied in question answering and knowledge-intensive tasks, its application to recruitment analytics and decision support remains limited. The Smart Recruitment Assistant positions itself at the intersection of these approaches by combining semantic matching with RAG-based interaction.

3 Methodology and Tools

3.1 Project Methodology

This academic project was conducted within the *Data Science* module following an iterative and modular development methodology. The objective was to design an intelligent recruitment system capable of processing heterogeneous and unstructured recruitment documents, extracting meaningful information, and supporting decision-making through explainable AI techniques.

The development process was organized into successive stages aligned with a Data Science pipeline: data acquisition, preprocessing, semantic representation, analysis, and interaction. Each stage was validated independently before integration, ensuring robustness while allowing incremental refinement of the recruitment-specific functionalities.

3.2 Data Science Workflow

The project follows a structured Data Science workflow applied to recruitment data:

1. **Data Ingestion**: Resumes and job descriptions are collected in multiple formats (PDF, DOCX, TXT), reflecting the diversity of real-world recruitment data.
2. **Data Preprocessing**: Raw documents are converted into clean textual representations. Quality checks, basic language detection, and metadata extraction are performed to assess the reliability of the extracted content.
3. **Semantic Information Extraction**: A Large Language Model is employed to extract structured information such as skills, professional experience, and education from unstructured text. This approach provides greater flexibility and robustness compared to traditional rule-based or Named Entity Recognition techniques.
4. **Feature Representation**: Textual data is transformed into numerical vector representations using sentence-level embeddings. These embeddings enable semantic similarity computation between resumes and job descriptions and support downstream retrieval tasks.
5. **Analysis and Decision Support**: Multi-dimensional matching algorithms compute interpretable scores, while Retrieval-Augmented Generation enables natural language interaction and explanation over the recruitment data.

This workflow ensures a clear separation between data preparation, modeling, and reasoning phases, which is a fundamental principle in applied Data Science projects.

3.3 Tools and Technologies

The implementation relies on a coherent and modern technology stack:

- **Frontend**: React with TypeScript is used to build interactive dashboards for recruiters and candidates, supporting visualization and real-time interaction.
- **Backend**: FastAPI is employed for API development, orchestration of processing pipelines, and input validation.
- **Artificial Intelligence and NLP**: Large Language Models (Gemini) are used for semantic extraction, summarization, and Retrieval-Augmented Generation, while transformer-based sentence embeddings support similarity computation.
- **Data Storage**: SQLite is used for structured relational data storage, and ChromaDB is used as a vector database to enable efficient semantic search and retrieval.

3.4 Justification of Design Choices

The choice of technologies and methodology was guided by both academic and practical considerations. React and FastAPI provide a clear separation between presentation and business logic, facilitating modular development and experimentation. SQLite offers simplicity, reliability, and reproducibility, making it well-suited for academic projects.

The use of a vector database such as ChromaDB enables the implementation of semantic retrieval and Retrieval-Augmented Generation without introducing unnecessary architectural complexity. Overall, the adopted methodology and tools support interpretability, extensibility, and reproducibility, which are key requirements for Data Science applications in an academic context.

4 System Architecture

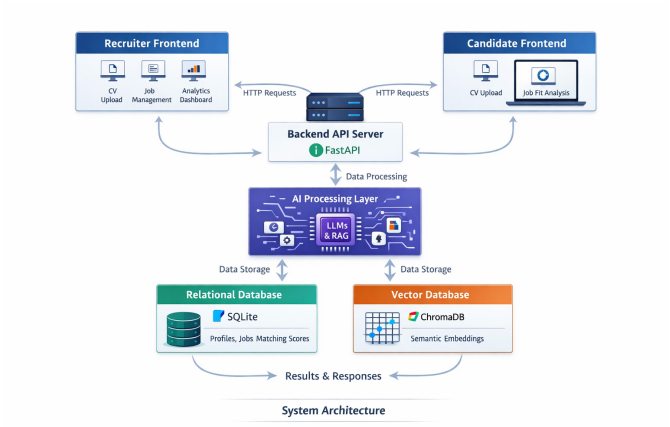

Figure 1. High-level architecture of the Smart Recruitment Assistant system

The Smart Recruitment Assistant is designed as a full-stack, modular system that combines modern web technologies with advanced artificial intelligence components. The architecture follows a layered approach to ensure separation of concerns, scalability, and maintainability. It integrates a frontend layer for user interaction, a backend layer for orchestration and business logic, an AI processing layer for semantic reasoning, and a data storage layer for structured and unstructured information.

4.1 High-Level Architecture Overview

At a high level, the system adopts a client–server architecture. The frontend, implemented using React and TypeScript, communicates with the backend through RESTful APIs exposed by a FastAPI server. The backend acts as the central coordinator, handling document uploads, triggering AI processing pipelines, managing data persistence, and serving responses to the frontend.

The AI layer is integrated within the backend and relies on external Large Language Models for semantic extraction, summarization, and reasoning tasks. Data persistence is handled through a hybrid storage approach, combining a relational database and a vector database.

4.2 Frontend Layer

The frontend layer is responsible for user interaction and data visualization. It provides two distinct dashboards corresponding to the two main user personas:

- **Recruiter Dashboard**: Enables recruiters to upload CVs and job descriptions, view matching results, analyze candidate pipelines, compare profiles, and interact with the AI assistant.
- **Candidate Dashboard**: Allows job seekers to upload their CVs and target job descriptions, view fit analysis, identify skill gaps, and receive personalized recommendations.

The frontend communicates exclusively with the backend via HTTP requests, ensuring that all business logic and AI processing remain centralized.

4.3 Backend Layer

The backend layer is implemented using FastAPI and serves as the core of the system. It exposes a set of REST API endpoints for document upload, matching, summarization, candidate management, and RAG-based queries.

Key responsibilities of the backend include:

- Handling file uploads and input validation
- Orchestrating document processing and matching pipelines
- Managing interactions with AI components
- Persisting structured data in the relational database
- Serving processed results to the frontend

4.4 AI Processing Layer

The AI processing layer encapsulates all intelligent functionalities of the system. It includes semantic information extraction using Large Language Models, semantic representation through embeddings, intelligent multi-dimensional matching, and a Retrieval-Augmented Generation pipeline enabling natural language querying with contextual grounding.

4.5 Data Storage Layer

The system adopts a hybrid data storage architecture:

- **Relational Database (SQLite)**: Stores structured data such as candidate profiles, job descriptions, and matching results.
- **Vector Database (ChromaDB)**: Stores semantic embeddings to support similarity search and Retrieval-Augmented Generation.

4.6 Data Flow and Interaction

The interaction between system components follows a well-defined data flow. Users upload documents through the frontend, the backend processes and analyzes the data using AI components, structured results are stored in the relational database, embeddings are indexed in the vector database, and final results are returned to the frontend for visualization and interaction.

5 Document Processing Pipeline

The effectiveness of the Smart Recruitment Assistant relies heavily on its ability to process heterogeneous and unstructured recruitment documents in a reliable and consistent manner. This section describes the document processing pipeline adopted in the project, which transforms raw resumes and job descriptions into structured and semantically meaningful representations suitable for analysis and reasoning.

The pipeline is designed following Data Science best practices, with a clear separation between data ingestion, preprocessing, information extraction, and semantic representation.

5.1 Document Ingestion

The pipeline begins with the ingestion of resumes and job descriptions provided by users through the frontend interface. The system supports multiple document formats, including PDF, DOCX, and TXT, in order to reflect real-world recruitment scenarios.

Uploaded files are validated at the backend level to ensure format compatibility and size constraints. Each document is associated with metadata such as filename, upload timestamp, and doc-

ument type (CV or job description), which facilitates traceability and downstream processing.

5.2 Text Extraction

Once a document is ingested, the system extracts its textual content using format-specific extraction techniques. For standard text-based PDF and DOCX files, dedicated libraries are used to retrieve the embedded text. In cases where PDF documents contain scanned images rather than selectable text, an Optical Character Recognition (OCR) fallback mechanism is employed.

This dual extraction strategy ensures robustness against document variability and minimizes information loss. The extracted text is normalized and stored as raw textual data, serving as the single source of truth for subsequent processing stages.

5.3 Data Preprocessing and Quality Control

Before semantic analysis, the extracted text undergoes preprocessing steps to improve quality and consistency. These steps include text normalization, removal of non-informative characters, and basic language detection. Additionally, metadata such as word count, character count, and extraction method are computed to assess document quality.

Documents that do not meet minimal quality thresholds can be flagged for review, ensuring that downstream AI components operate on reliable input data.

5.4 Semantic Information Extraction

To convert unstructured text into structured representations, the system employs a Large Language Model for semantic information extraction. Instead of relying solely on traditional rule-based or Named Entity Recognition techniques, the model is prompted to extract relevant entities such as skills, professional experience, education, and key qualifications.

The extracted information is returned in a structured JSON format, enabling seamless integration with the relational database and analytical modules. This approach improves adaptability to diverse document layouts and writing styles commonly encountered in recruitment data.

5.5 Semantic Representation and Indexing

Following information extraction, textual data is transformed into vector representations using sentence-level embeddings. These embeddings capture the semantic meaning of documents and enable similarity-based analysis.

The generated embeddings are indexed in a vector database after chunking documents into overlapping segments, allowing fine-grained semantic search and supporting Retrieval-Augmented Generation.

5.6 Pipeline Integration

The document processing pipeline is fully integrated into the backend architecture and operates automatically upon document upload. Structured outputs are stored in the relational database, while semantic representations are persisted in the vector database.

This hybrid approach provides a reliable foundation for intelligent matching, analytics, and natural language interaction within the Smart Recruitment Assistant.

6 Intelligent CV–Job Matching

A core component of the Smart Recruitment Assistant is its intelligent CV–Job matching mechanism. Unlike traditional recruitment systems that rely on keyword-based filtering, the proposed approach performs a multi-dimensional semantic analysis to evaluate the alignment between candidate profiles and job requirements in an interpretable and explainable manner.

The matching process combines semantic similarity with explicit domain-specific criteria, enabling a more accurate and fair assessment of candidate suitability.

6.1 Overview of the Matching Strategy

The matching strategy is designed to reflect how recruiters evaluate candidates in real-world scenarios. Rather than relying on a single similarity score, the system computes multiple complementary scores, each capturing a different aspect of candidate–job alignment.

The final match score is obtained through a weighted aggregation of these components, allowing fine-grained analysis and transparency in decision-making.

6.2 Multi-Dimensional Scoring Model

The overall matching score is computed as a weighted combination of four main components:

- **Semantic Similarity (40%)**: Measures the overall semantic alignment between the CV and the job description.
- **Skills Matching (30%)**: Evaluates the overlap between candidate skills and required job skills.
- **Experience Matching (20%)**: Assesses whether the candidate's professional experience satisfies the job's experience requirements.
- **Education Matching (10%)**: Compares the candidate's educational background with the job's educational expectations.

This weighting scheme balances semantic understanding with explicit recruitment criteria, ensuring both flexibility and interpretability.

6.3 Semantic Similarity Computation

Semantic similarity is computed by transforming both the CV and the job description into vector representations using sentence-level embeddings. The similarity between these vectors is calculated using cosine similarity, enabling the detection of contextual similarities beyond exact word overlap.

6.4 Skills Matching

Skills matching is performed using a hybrid strategy combining exact matching, fuzzy matching, and semantic similarity. This approach identifies both directly matched skills and semantically related competencies, resulting in lists of matched and missing skills.

6.5 Experience and Education Matching

Experience matching compares the candidate's total years of professional experience with the minimum experience required for the job, allowing partial satisfaction when appropriate. Education matching evaluates degree level and field relevance, contributing additional structure to the final score.

6.6 Interpretability and Result Generation

The system emphasizes interpretability by providing an overall match score, a letter grade, detailed score breakdowns, identified strengths and weaknesses, and personalized recommendations. This transparency supports informed decision-making for both recruiters and candidates.

6.7 Integration with the Decision Support System

Matching results are stored in the relational database and reused across analytics dashboards, candidate comparison tools, and Retrieval-Augmented Generation queries, ensuring consistency and coherence across system functionalities.

7 Retrieval-Augmented Generation (RAG)

To enhance explainability and interaction beyond traditional scoring mechanisms, the Smart Recruitment Assistant integrates a Retrieval-Augmented Generation (RAG) framework. This approach enables the system to answer natural language questions about candidates and job descriptions by grounding generative responses in retrieved, relevant document content.

By combining information retrieval with Large Language Models, the RAG component transforms the recruitment platform into an interactive decision-support system.

7.1 Motivation for Using RAG

While semantic matching provides quantitative insights into candidate–job alignment, recruiters and candidates often require qualitative explanations and comparative reasoning. Simple similarity scores are insufficient to answer questions such as why a candidate is suitable or how candidates differ.

RAG addresses this limitation by retrieving relevant document segments and conditioning the language model on this context, ensuring that generated responses are both informative and grounded in actual data.

7.2 RAG Architecture Overview

The RAG pipeline follows a retrieve-and-generate paradigm:

1. A user submits a natural language query.
2. The system identifies the appropriate query context.
3. Relevant document chunks are retrieved from a vector database using semantic similarity.
4. The retrieved content is combined with the user query to form a contextual prompt.
5. A Large Language Model generates a response based on this grounded context.

This architecture reduces hallucinations and improves answer reliability.

7.3 Document Chunking and Vector Indexing

To support efficient retrieval, CVs and job descriptions are segmented into overlapping text chunks prior to indexing. Each chunk is transformed into a vector embedding and stored in a vector database along with metadata such as document type and candidate identifier.

This indexing strategy enables fine-grained semantic retrieval and supports complex queries.

7.4 Query Modes and Context Selection

The system supports multiple RAG query modes:

- **Specific Candidate Mode**: Queries related to a single candidate's profile.
- **Job-Specific Mode**: Queries comparing multiple candidates for a given job opening.
- **Global Database Mode**: Queries spanning the entire candidate database.

Dynamic context selection ensures that responses remain relevant and focused.

7.5 Persona-Aware Response Generation

The RAG system adapts its responses based on the user persona. Recruiter-oriented responses emphasize evaluation and comparison, while candidate-oriented responses focus on self-assessment and improvement recommendations.

7.6 Source Attribution and Explainability

Each generated response is accompanied by explicit source references corresponding to the retrieved document segments. This design enhances transparency, trust, and explainability in recruitment decision-making.

7.7 Integration with the Overall System

The RAG component is integrated with matching and data management modules, enabling hybrid reasoning over structured and unstructured data. This integration extends the platform from a matching system to an intelligent conversational assistant.

8 Frontend and User Experience

The Smart Recruitment Assistant was designed with a strong emphasis on usability and clarity, ensuring that complex AI-driven functionalities are accessible to end users. The frontend plays a crucial role in translating analytical results into intuitive visual representations and interactive workflows.

The user experience is tailored to two distinct personas—recruiters and candidates—each benefiting from dedicated interfaces aligned with their specific objectives.

8.1 Frontend Architecture

The frontend is implemented using React with TypeScript, enabling the development of a modular, maintainable, and type-safe user interface. The application follows a component-based architecture, where each feature is encapsulated within reusable components.

Communication with the backend is handled through RESTful API calls, ensuring a clear separation between presentation logic and business logic. This design enhances scalability and simplifies future extensions.

8.2 Recruiter Dashboard

The recruiter dashboard supports decision-making throughout the recruitment pipeline. It enables recruiters to upload and manage CVs and job descriptions, view detailed match scores and breakdowns, access visual analytics, compare candidates side by side, and interact with the RAG-powered assistant.

The interface emphasizes interpretability by presenting both quantitative scores and qualitative insights, allowing recruiters to understand not only who matches best, but also why.

8.3 Candidate Dashboard

The candidate dashboard focuses on self-assessment and career development. It allows job seekers to analyze their fit for a target position, identify skill gaps, receive personalized recommendations, and interact with an AI career coach using natural language queries.

This design transforms the recruitment system into an active guidance tool rather than a passive evaluation mechanism.

8.4 Data Visualization and Feedback

The frontend integrates multiple visualization components to enhance result comprehension. Match scores are displayed using progress indicators and charts, while skill analysis and comparisons are presented through structured visual elements.

Consistent visual design and concise feedback reduce cognitive load and improve interpretability.

8.5 Responsiveness and Usability

The frontend follows a responsive design approach to ensure usability across different devices and screen sizes. Additional usability features include loading indicators, clear error messages, intuitive navigation, and markdown-formatted AI responses.

These design choices contribute to a smooth and user-friendly interaction with the system.

9 Experimental Evaluation & Use Cases

This section presents an experimental evaluation of the Smart Recruitment Assistant, focusing on the effectiveness, coherence, and usability of the proposed system. Given the academic nature of the project and its focus on applied Data Science, the evaluation combines quantitative analysis of matching results with qualitative assessment of system behavior and user interaction.

9.1 Experimental Setup

The system was evaluated using a small-scale dataset composed of resumes and job descriptions collected for academic experimentation. The dataset includes multiple CVs with varying levels of experience and skill sets, as well as several job descriptions targeting different technical profiles.

All experiments were conducted in a local development environment. The evaluation scenarios were designed to reflect realistic recruitment workflows, including recruiter-driven candidate selection and candidate self-assessment.

The objective of the evaluation is not to benchmark against industrial systems, but to verify the correctness, interpretability, and practical usefulness of the proposed approach.

9.2 CV–Job Matching Results

The intelligent matching module produces an overall match score expressed as a percentage, along with a letter grade ranging from A+ to D. The results show a coherent distribution of scores across candidates, reflecting differences in skill alignment, experience level, and semantic relevance.

High-ranking candidates typically exhibit strong semantic similarity with the job description and a high overlap of required skills. Lower scores correspond to profiles with missing key competencies or insufficient experience.

9.3 Skill Matching and Gap Analysis

The system generates detailed skill-level analysis for each candidate, explicitly identifying matched skills and missing skills. This information supports fine-grained assessment of candidate suitability and enables actionable feedback for profile improvement.

9.4 RAG-Based Question Answering Evaluation

The Retrieval-Augmented Generation component was evaluated qualitatively through natural language queries posed by recruiters and candidates. The generated responses were contextually relevant and grounded in retrieved document content.

Source attribution accompanying each response enhances transparency and trust, allowing users to verify the origin of the information.

9.5 Performance Analysis

The system demonstrates near real-time performance for core functionalities, including document parsing, matching computation, and RAG queries. These response times are suitable for interactive usage in academic and small-scale deployment contexts.

9.6 Use Case Scenarios

Two primary use cases were evaluated:

- **Recruiter Use Case**: Uploading job descriptions and CVs, analyzing match scores, comparing candidates, and interacting with the AI assistant.

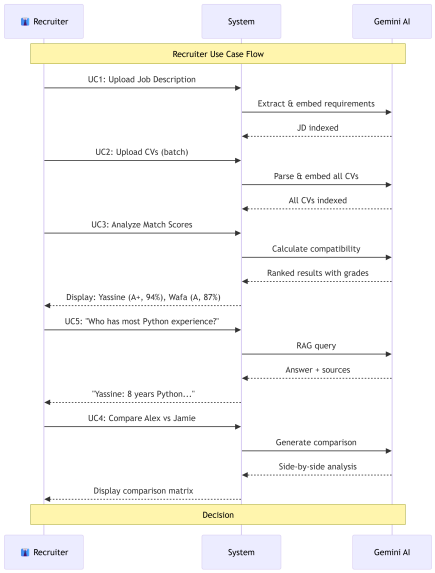

Figure 2. Recruiter Complete Use Case

- **Candidate Use Case**: Uploading CVs and job descriptions, receiving fit analysis, identifying skill gaps, and obtaining personalized recommendations.

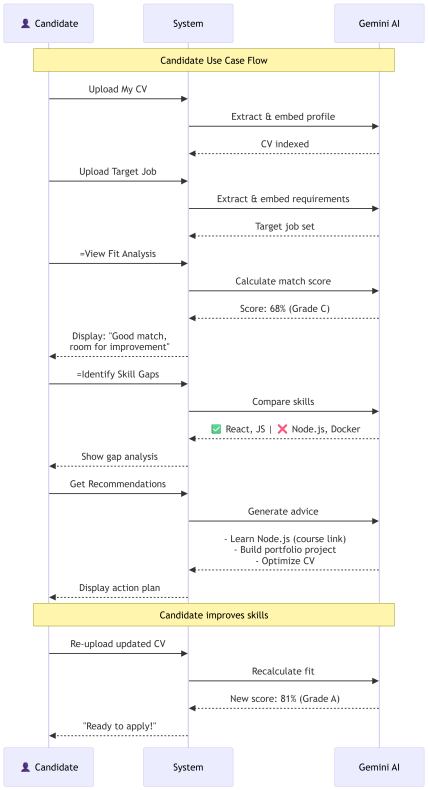

Figure 3. Candidate Use Case

10 Discussion and Limitations

This section discusses the strengths of the proposed Smart Recruitment Assistant and highlights its current limitations. As an academic project developed within the *Data Science* module, the system prioritizes methodological clarity, interpretability, and practical relevance over large-scale industrial deployment.

10.1 Discussion

The experimental evaluation demonstrates that the Smart Recruitment Assistant effectively combines semantic analysis, structured matching, and interactive reasoning to support recruitment-related decision-making. By integrating Large Language Models with vector-based retrieval and structured data analysis, the system moves beyond traditional keyword-based recruitment tools.

One of the key strengths of the proposed approach lies in its multi-dimensional and interpretable matching strategy. Rather than producing a single opaque score, the system provides a detailed breakdown of semantic similarity, skills alignment, experience, and education. This transparency enhances trust and enables both recruiters and candidates to understand the rationale behind matching results.

Another important contribution is the integration of Retrieval-Augmented Generation as an interaction mechanism. The RAG component allows users to query candidate data using natural language while ensuring that responses remain grounded in retrieved document content. This transforms the platform into an intelligent decision-support system.

From a Data Science perspective, the project demonstrates the effective combination of unstructured text processing, semantic representation, and analytical reasoning within a unified pipeline. The hybrid use of relational and vector databases further enhances flexibility and extensibility.

10.2 Limitations

Despite its strengths, the proposed system presents several limitations.

First, the system relies on external Large Language Model APIs for semantic extraction, summarization, and RAG-based reasoning. Consequently, API usage quotas and rate limits may restrict the number of documents processed or queries executed, particularly in academic or free-tier environments. In production settings, this limitation could be mitigated through paid plans, request caching, batching strategies, or the adoption of open-source language models.

Second, the experimental evaluation was conducted on a limited dataset suitable for academic experimentation. Larger-scale evaluations would be required to assess performance and robustness in real-world recruitment scenarios.

Third, the current implementation assumes a single primary language for document processing, limiting applicability in multilingual recruitment contexts.

Finally, the system does not currently include authentication, role-based access control, or explicit bias mitigation mechanisms, which are essential for deployment in real-world environments but fall outside the scope of this academic project.

10.3 Summary of Limitations

The main limitations of the system can be summarized as follows:

- Dependency on external LLM APIs and associated quota constraints
- Evaluation on a small-scale academic dataset
- Limited multilingual support
- Absence of authentication and bias control mechanisms

11 Conclusion

This paper presented **Smart Recruitment Assistant**, an AI-driven recruitment platform developed as an academic project within the *Data Science* course. The objective of the project was to design and implement an intelligent system capable of processing unstructured recruitment data, extracting meaningful information, and supporting decision-making through explainable and interactive AI techniques.

Throughout the project, core Data Science concepts were applied in a coherent and practical manner, including data preprocessing, semantic representation, similarity analysis, and hybrid reasoning over structured and unstructured data. By combining multi-dimensional CV–Job matching with Retrieval-Augmented Generation, the system goes beyond traditional keyword-based recruitment tools and provides both quantitative and qualitative insights.

The proposed architecture demonstrates how modern Data Science pipelines can be operationalized using a full-stack approach that integrates machine learning models, vector databases, and web technologies. The system effectively supports two distinct user personas—recruiters and candidates—by offering tailored functionalities such as interpretable scoring, skill gap analysis, and natural language interaction.

As an academic project, the Smart Recruitment Assistant illustrates the practical application of Data Science methods to a real-world problem. While the system presents certain limitations, including reliance on external LLM APIs and small-scale evaluation, it establishes a solid foundation for future enhancements such as multilingual support, larger-scale validation, and deployment-oriented features.

In conclusion, this project highlights the potential of combining Data Science, natural language processing, and modern AI paradigms to improve transparency, efficiency, and usability in recruitment processes, while serving as a comprehensive learning experience within the Data Science curriculum.

Acknowledgements

This project received support during the Data Science course, instructed by Professor MAAROUF Otman, PhD at the Faculty of Sciences, Ibn Zohr University.