TSYP CS Challenge

Technical Report

AI Career Assistance Platform

Contents

Chapter 1

Introduction and Goals

This report presents the technical design and implementation details of the **AI Career Assistance Platform**, a modular and scalable system built to help users improve employability through AI-powered features such as CV optimization, job matching, and interview preparation.

System Overview

The platform is composed of five main services:

- **AI Interviewer**: Simulates and evaluates interview sessions using LLM-based questioning and scoring.
- **Job Matcher**: Scrapes, filters, and recommends relevant job postings using semantic similarity.
- **CV Reviewer**: Analyses user CVs, computes scores (quality, relevance, completeness), and generates personalized feedback.
- **CV Rewriter**: Generates improved and standardized CV versions.

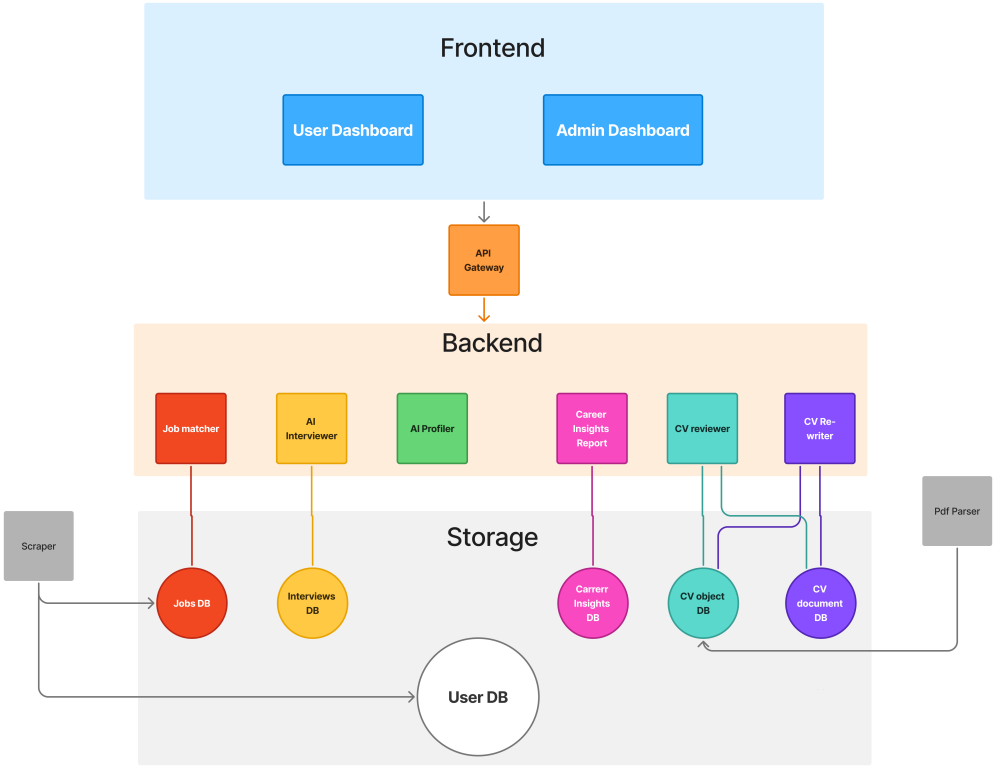

Figure 1.1: Global Architecture

Chapter 2

Technologies Used

Layer	Technologies
Backend	Flask, Python, LangChain
Frontend	React
Database	ChromaDB (CV & embeddings),FireStore
AI Layer	LLMs: nvidia/nemotron-nano-9b-v2, alibaba/tongyi-deepresearch-30b-a3b, Google Gemini-2 Embeddings: sentence-transformers/all-MiniLM-L6-v2, intfloat/e5-large-v2 Whisper & TTS: GTTS, pygame
Storage	Google Cloud Storage
Deployment	Google Cloud (GCP), Docker
NLP Libraries	LanguageTool, textstat

Chapter 3

CV Reviewer & Rewriter

3.1 CV Extraction System

Goal

Automatically extract structured information from user CVs, regardless of format or layout.

Extracted Fields

- Personal information (name, email, phone)
- Social links (LinkedIn, GitHub, portfolio)
- Professional summary
- Education (degree, school, dates, GPA)
- Experience (company, title, responsibilities, achievements)
- Projects, awards, certifications, volunteering, publications
- Skills (technical, soft, languages), hobbies & interests

Approaches Evaluated

Approach	Advantages	Disadvantages
Traditional NLP (Regex, NLTK, spaCy)	Fast, deterministic, low-cost, offline	Fragile to CV layouts, poor semantic understanding
RAG-based Extraction (LLM + Retrieval)	Excellent semantic extraction, works on any format	Slower, requires online LLM, higher cost

Choice: RAG-based extraction due to semantic accuracy and robustness.

3.2 RAG Pipeline

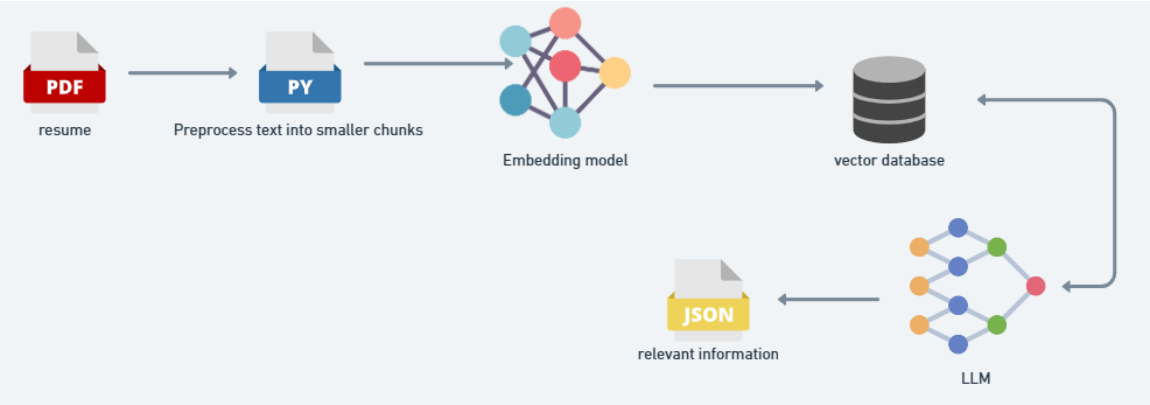

Figure 3.1: RAG-based CV Extraction Pipeline

3.2.1 Embedding Model

`HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")`

3.2.2 PDF Loading & Chunking

```
loader = PyPDFLoader(DATA_PATH)
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000, chunk_overlap=200
)
```

3.2.3 Vector Store – ChromaDB

All chunks are embedded and persisted in a Chroma collection.

3.2.4 Retriever

```
retriever = vector_store.as_retriever(search_kwargs={"k": 7})
```

3.2.5 LLM Extraction with Fallback

LLM used: `alibaba/tongyi-deepresearch-30b-a3b`

Example Prompt

```
You are an expert extraction algorithm.
Return structured JSON. If a field is unknown, return null.

Focus on:
{"personal_information": {
    "full_name": "",
    "email": "",
    "phone": ""
}}
```

Fallback Steps

- clean_text()
- repair_json()

3.3 CV Reviewer

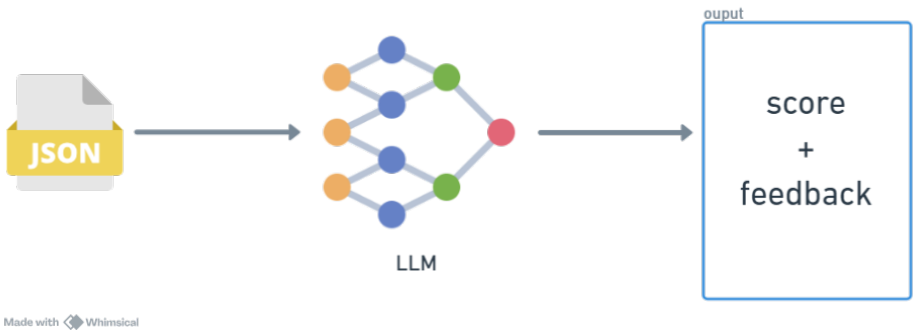

Figure 3.2: CV Reviewer Flow

Step 1 — Loading & Preprocessing

The CV is parsed from its JSON structure.

Step 2 — Relevance Score

Evaluates alignment with job title: skills fit, experience relevance, achievements.

Step 3 — Quality Score

Uses:

- LanguageTool grammar checks
- Readability (textstat)
- Structural penalties

Step 4 — Section Completeness

Mandatory: Experience, Education, Skills, Projects, Social links, Personal info.

Step 5 — Final Score

$$CV\,Score = 0.4Relevance + 0.4Section + 0.2Quality$$

3.4 CV Rewriter

3.4.1 User Inputs & Outputs

Inputs: PDF (text or scanned). **Outputs:** DOCX templates (Europass, Canadian, Minimalist).

3.4.2 Rewriting Prompt

```
SYSTEM: You are a CV-writing expert.
USER: Rewrite the following section in professional French:
{original_section}
```

3.4.3 Example JSON Output

```
{
  "section": "experience",
```

```
  "input": "...",
  "output": "...",
  "validation": {"coherence": true, "grammaire": 0.98}
}
```

Chapter 4

Job Matching System

4.1 Goal

Match user CVs with the most relevant job descriptions using semantic retrieval.

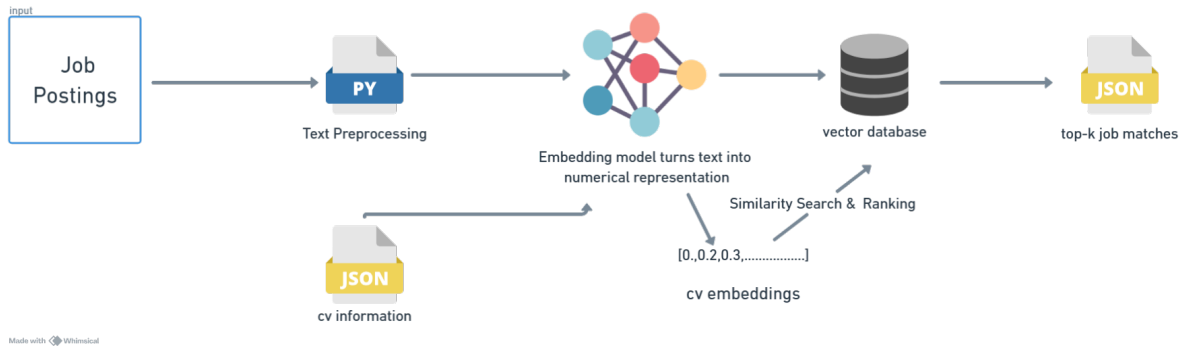

Figure 4.1: Job Matching RAG Pipeline

4.2 Embedding Model

we used the embedding model `intfloat/e5-large-v2` because The model performs well on both short and long text inputs and it is pretrained for semantic similarity and information retrieval tasks.Additionally, it offers a good balance between performance and computational efficiency,

4.3 Vector Store

Stored with metadata:

- job_id
- title
- company
- location

4.4 Search & Ranking

1. Compute CV embedding
2. Perform similarity search
3. Retrieve top 10 jobs
4. Rank by semantic similarity

Chapter 5

AI Interviewer

5.1 Main Objectives

- Simulate recruiter interviews
- Analyze quality of responses
- RAG-based context from CV
- Maintain dialogue memory

5.2 Interview Flow

Initialization

Load Gemini LLM, vectorize CV into ChromaDB.

Interview Loop

1. Record audio (Whisper)
2. Analyze response: empty, vague, confused, coherent
3. Retrieve CV context
4. Generate next question
5. Read aloud using GTTS

5.3 RAG Module

- Semantic retrieval of CV chunks (k=2–7)
- Ensures question coherence

5.4 Response Analysis

- Empty → reformulate
- Vague → simplify question
- Confused → clarify gently
- Coherent → move to next topic

5.5 Expected Results

- Natural, interactive interview simulation
- CV-based intelligent questioning

- Automatic clarifications

Conclusions

The **AI Career Assistance Platform** demonstrates a modular, AI-driven approach to enhancing employability. By combining RAG pipelines, semantic embeddings, and advanced LLMs, it can extract structured CV data, assess and improve CV quality, match candidates with relevant jobs, and simulate intelligent interviews.

The system balances accuracy, efficiency, and scalability, highlighting the power of modern NLP and retrieval-based AI techniques.