AI-Powered Job Screening Backend Service

Overview

This project implements a backend API that automates the evaluation of job applications using AI. The system receives a candidate’s CV and project report, compares them against predefined job descriptions and rubrics, and produces a structured evaluation. The service is built with **FastAPI**, uses **Redis Stack** for vector similarity search, and integrates **Gemini 1.5 Flash** as the LLM. Embeddings are generated using **BAAI/bge-m3 (safetensors)** to enable contextual RAG retrieval.

System Architecture

1. **Upload Module (/upload)**: Accepts candidate CV and project report, stores with unique IDs.
2. **Evaluation Pipeline (/evaluate)**: Runs asynchronously, retrieves context from Job Description, Case Study Brief, and Rubrics. Generates structured evaluation using LLM.
3. **Result Retrieval (/result/{id})**: Returns job status and final structured result.

Key Components

Component	**Technology**
API Framework	FastAPI
Vector DB	Redis Stack (HNSW)
Embedding Model	BAAI/bge-m3
LLM	Gemini 1.5 Flash
Language	Python 3.11
Job Handling	BackgroundTasks (future: Redis Queue)

Prompt Design Example

CV Evaluation Prompt:
Evaluate the CV against the job description and scoring rubric.
Return JSON: {"cv_match_rate": float, "cv_feedback": string}

Project Evaluation Prompt:
Evaluate the project report for correctness, code quality, and RAG implementation.
Return JSON: {"project_score": float, "project_feedback": string}

Reflection

This system successfully integrates LLM evaluation into a backend service with minimal latency. Future improvements include moving job storage to Redis Queue, adding retry/backoff handling, and deploying via Docker for reproducibility.