Case Study Brief — Backend Developer Technical Assignment

1. Overview

Terima kasih telah melamar sebagai **Backend Developer**.

Tugas mini project ini harus diselesaikan dalam waktu **5 hari** sejak Anda menerima dokumen ini.

Tujuan dari proyek ini adalah untuk menilai kemampuan Anda dalam **menggabungkan backend engineering dengan AI workflows**, termasuk prompt design, LLM chaining, retrieval, dan resilience.

---

2. Objective

Misi Anda adalah membangun **backend service** yang mengotomatisasi proses **screening awal lamaran pekerjaan**.

Service ini akan menerima **CV** dan **Project Report** dari kandidat, mengevaluasinya terhadap **Job Description** dan **Case Study Brief**, lalu menghasilkan **structured AI-generated evaluation report**.

---

3. Core Logic & Data Flow

3.1 Candidate-Provided Inputs (Data yang Dinilai)

1. **Candidate CV** — File PDF berisi resume kandidat.
2. **Project Report** — File PDF laporan hasil pengerjaan case study kandidat.

3.2 System-Internal Documents (Ground Truth)

1. **Job Description** — Dokumen berisi deskripsi pekerjaan yang menjadi acuan evaluasi CV.
    - Anda dapat menggunakan deskripsi pekerjaan dari posisi yang Anda lamar.
    - Untuk memastikan akurasi retrieval, disarankan meng-ingest beberapa job description.
2. **Case Study Brief** — Dokumen ini, digunakan sebagai acuan evaluasi Project Report.
3. **Scoring Rubric** — Sekumpulan parameter penilaian untuk CV dan Project Report (masing-masing dalam PDF terpisah).

---

4. Deliverables

4.1 Backend Service (API Endpoints)

Bangun layanan backend dengan minimal tiga endpoint berikut.

a. POST /upload

- Menerima `multipart/form-data` yang berisi:
  - Candidate CV (PDF)
  - Project Report (PDF)
- Menyimpan file dan mengembalikan ID unik untuk masing-masing.

Response Example

- {
- "cv_id": "123",
- "report_id": "456"
- }
-

b. POST /evaluate

- Memicu proses **AI evaluation pipeline** secara **asynchronous**.
- Input:
  - `job_title` (string)
  - `cv_id`
  - `report_id`
- Mengembalikan `job_id` untuk tracking proses evaluasi.

Response Example

- {
- "id": "456",
- "status": "queued"
- }
-

c. GET /result/{id}

- Mengambil status dan hasil evaluasi berdasarkan `job_id`.

While queued or processing

- {

- "id": "456",
- "status": "queued" | "processing"
- }
-

Once completed

- {
- "id": "456",
- "status": "completed",
- "result": {
- "cv_match_rate": 0.82,
- "cv_feedback": "Strong in backend and cloud, limited AI integration experience...",
- "project_score": 4.5,
- "project_feedback": "Meets prompt chaining requirements, lacks error handling robustness...",
- "overall_summary": "Good candidate fit, would benefit from deeper RAG knowledge..."
- }
- }
-

---

5. Evaluation Pipeline

Pipeline AI dijalankan ketika endpoint `/evaluate` dipanggil, dan mencakup beberapa komponen utama.

5.1 RAG (Context Retrieval)

- Ingest seluruh **System-Internal Documents** (Job Description, Case Study Brief, Scoring Rubrics) ke dalam **vector database**.
- Retrieve bagian relevan untuk dimasukkan ke dalam prompt sesuai konteks evaluasi (misalnya: “for CV scoring” vs “for project scoring”).

5.2 Prompt Design & LLM Chaining

Pipeline evaluasi terdiri dari tiga tahap utama:

a. CV Evaluation

- Parse CV kandidat menjadi data terstruktur.
- Retrieve informasi relevan dari **Job Description** dan **CV Scoring Rubric**.
- Gunakan LLM untuk menghasilkan:

    - `cv_match_rate`
    - `cv_feedback`

b. Project Report Evaluation

- Parse Project Report menjadi data terstruktur.
- Retrieve informasi relevan dari **Case Study Brief** dan **Project Scoring Rubric**.
- Gunakan LLM untuk menghasilkan:
    - `project_score`
    - `project_feedback`

c. Final Analysis

- Gunakan satu kali panggilan LLM tambahan untuk mensintesis hasil dari tahap sebelumnya menjadi:
    - `overall_summary`

---

6. Long-Running Process Handling

- Endpoint `POST /evaluate` tidak boleh menunggu hingga proses selesai.
- Simpan task dan kembalikan `job_id` agar hasil dapat diambil melalui `GET /result/{id}` secara berkala.
- Pastikan pipeline dapat menangani tugas yang berjalan lama secara efisien.

---

7. Error Handling & Randomness Control

- Simulasikan berbagai edge case, seperti kegagalan parsing atau input tidak valid.
- Tangani kemungkinan error dari API LLM (misalnya timeout, rate limit).
- Implementasikan **retry/back-off mechanism**.
- Kontrol parameter LLM seperti temperature untuk menjaga stabilitas hasil.
- Tambahkan lapisan validasi untuk memastikan hasil evaluasi konsisten.

---

8. Standardized Evaluation Parameters

8.1 CV Evaluation (Match Rate)

Parameter	Deskripsi

Technical Skills Match	Kesesuaian keterampilan teknis (backend, database, API, cloud, AI/LLM).
Experience Level	Lama pengalaman dan kompleksitas proyek yang pernah dikerjakan.
Relevant Achievements	Dampak dan skala pencapaian yang relevan.
Cultural Fit	Sikap kerja, komunikasi, dan semangat belajar.

8.2 Project Deliverable Evaluation

Parameter	Deskripsi
Correctness	Kesesuaian dengan requirements (prompt design, chaining, RAG, error handling).
Code Quality	Struktur kode yang bersih, modular, dan mudah diuji.
Resilience	Kemampuan menangani kegagalan dan melakukan retry.
Documentation	Kualitas README dan penjelasan trade-off.
Creativity / Bonus	Fitur tambahan seperti autentikasi, deployment, atau dashboard.

Nilai tiap parameter: **1–5**, kemudian dirata-rata untuk menghasilkan skor akhir.

---

9. Technical Requirements

1. Gunakan framework backend apa pun (mis. Django, Node.js, Rails, dll).
2. Gunakan layanan LLM seperti OpenAI, Gemini, atau OpenRouter (boleh menggunakan versi gratis).
3. Gunakan vector database sederhana seperti **ChromaDB**, **Qdrant**, atau layanan RAG-as-a-service.
4. Sediakan **README** dengan instruksi instalasi dan penjelasan desain.
5. Sertakan seluruh dokumen internal serta skrip ingestion di dalam repositori agar hasil dapat direproduksi.