AI-Powered Recruitment Assistant - Backend Developer Case Study

Case Study Brief

Objective

Build an AI-powered recruitment assistant system for evaluating backend developer candidates. The system should automatically analyze CVs and project submissions to assess candidate fit against job requirements, using standardized rubrics and semantic search capabilities.

Context/Background

The recruitment process for backend developers requires careful evaluation of both technical expertise (backend frameworks, databases, APIs, cloud, AI/LLM) and practical implementation skills. Manual review of CVs and technical projects is time-consuming and prone to inconsistency. This system aims to standardize and automate the initial screening phase while maintaining high-quality candidate assessment.

Core Logic & Data Flow

1. **Document Ingestion**: Upload and parse CVs (PDF format) and extract structured information
2. **Semantic Indexing**: Store job descriptions, evaluation rubrics, and case study briefs in a vector database (ChromaDB) for semantic retrieval
3. **CV Evaluation**: Compare candidate profiles against job requirements using AI-driven semantic matching
4. **Project Assessment**: Evaluate technical deliverables based on code quality, correctness, and documentation
5. **Scoring & Ranking**: Generate standardized scores and provide detailed feedback for each candidate

Deliverables

Backend Service - RESTful API endpoints for CV upload and evaluation - Integration with cloud storage (AWS S3) for document management - Database layer (PostgreSQL + Prisma ORM) for persisting evaluation results - Authentication and authorization mechanisms

Evaluation Pipeline - AI-powered CV parsing and information extraction - Semantic search using ChromaDB with Google Gemini embeddings - Automated scoring based on predefined rubrics - Support for both CV evaluation and project deliverable assessment

Standardized Evaluation Parameters - CV Match Evaluation: Technical skills, experience level, achievements, cultural fit - Project Evaluation: Code

quality, correctness, error handling, documentation, creativity

Requirements

Functional Requirements - Upload CVs in PDF format - Extract candidate information (skills, experience, education, projects) - Evaluate CVs against job description using semantic similarity - Score candidates based on standardized rubrics - Store and retrieve evaluation results - Generate detailed evaluation reports

Non-Functional Requirements - Response time $<$ 30 seconds for CV evaluation - Support concurrent evaluations - Secure handling of candidate data - Scalable architecture for batch processing - Comprehensive error handling and logging

Technical Stack

Backend - Runtime: Node.js with NestJS framework - Language: TypeScript - Database: PostgreSQL with Prisma ORM - Vector Database: ChromaDB for semantic search - Cloud Storage: AWS S3 for document storage - AI/LLM: Google Gemini API for embeddings and content generation

Development Tools - Package Manager: pnpm - API Testing: Postman/REST Client - Environment Management: dotenv - Code Quality: ESLint, Prettier

---

Job Description - Product Engineer (Backend) 2025

About the Job

We are seeking a Product Engineer (Backend) to build AI-powered systems and backend features. You will be responsible for developing backend solutions, integrating AI/LLM workflows, and ensuring our applications are robust and scalable.

Key Responsibilities

- Collaborate with frontend engineers to build robust backend solutions
- Develop and maintain server-side logic for central database
- Design and fine-tune AI prompts that align with product requirements
- Build LLM chaining flows and implement RAG using vector databases
- Handle long-running AI processes with job orchestration and retry mechanisms
- Implement safeguards for API failures and LLM output randomness
- Write reusable, testable, and efficient code
- Conduct full product lifecycles from design to deployment

About You

Required Skills & Experience - 3+ years of backend development experience - Experience with backend frameworks (Node.js, Django, Rails) - Database management (MySQL, PostgreSQL, MongoDB) - RESTful APIs and cloud technologies (AWS, Google Cloud, Azure) - Understanding of authentication, authorization, and scalable design - Creating database schemas and implementing automated testing

Preferred Qualifications - Familiarity with LLM APIs, embeddings, vector databases - Experience with prompt design and AI workflow orchestration - Knowledge of RAG (Retrieval-Augmented Generation) - Experience handling async processes and background jobs

Benefits & Perks

- Competitive salary and flexible remote work
- 17 days paid time off per year
- Learning budget (Rp29 million/year) for courses and resources
- Device ownership budget (Rp7 million/year)
- Health insurance and professional development support

CV Match Evaluation Rubric

Parameter 1: Technical Skills Match (Weight: 40%)

Description: Evaluate the candidate's backend-specific technical skills against job requirements, including backend frameworks, databases, APIs, cloud, and AI/LLM exposure.

Scoring Guide: - **5 (Excellent)**: Expert-level proficiency in backend frameworks, databases, and cloud. Deep understanding of AI/LLM integration, RAG, and prompt engineering. Active contributions to open-source projects. - **4 (Good)**: Strong proficiency in backend development with solid database and API experience. Experience with cloud platforms and some AI/LLM integration work. - **3 (Satisfactory)**: Moderate proficiency in backend frameworks and databases. Familiar with RESTful APIs and basic cloud deployment. Limited AI/LLM experience. - **2 (Below Average)**: Basic backend knowledge with minimal production experience. Limited exposure to cloud or AI technologies. - **1 (Poor)**: No relevant backend development experience. Lacks fundamental knowledge of databases, APIs, or backend architecture.

Parameter 2: Experience Level (Weight: 25%)

Description: Assess the candidate's years of relevant experience in backend development and overall software engineering background.

Scoring Guide: - **5 (Excellent)**: 5+ years of backend development with high-impact projects. Led multiple successful projects from conception to production. - **4 (Good)**: 3-4 years of backend development with solid track record. Contributed to several production projects with full development lifecycle experience. - **3 (Satisfactory)**: 2-3 years of backend development with mid-scale projects. Growing expertise with some production experience. - **2 (Below Average)**: Less than 2 years backend development. Limited project experience, primarily junior-level work. - **1 (Poor)**: No backend development experience or less than 1 year total software development.

Parameter 3: Relevant Achievements (Weight: 20%)

Description: Evaluate the candidate's track record of achievements in backend development, including deployed systems, contributions to notable projects, or certifications.

Scoring Guide: - **5 (Excellent)**: Multiple deployed backend systems with significant impact. Published technical articles or research. Recognized contributor to major open-source projects. - **4 (Good)**: Several production backend systems deployed. Active open-source contributions. Relevant certifications or hackathon participation. - **3 (Satisfactory)**: At least one deployed backend system. Some involvement in tech community. Basic portfolio of projects. - **2 (Below Average)**: Limited demonstrated achievements. Mostly personal or learning projects. - **1 (Poor)**: No demonstrable backend achievements. No portfolio or community involvement.

Parameter 4: Cultural/Collaboration Fit (Weight: 15%)

Description: Assess the candidate's alignment with team values, communication skills, collaborative mindset, and passion for technology and learning.

Scoring Guide: - **5 (Excellent)**: Exceptional communication skills with clear passion for technology. Strong evidence of collaborative work. Demonstrates leadership and mentoring abilities. - **4 (Good)**: Good communication skills and genuine interest in technology. Clear examples of effective teamwork. - **3 (Satisfactory)**: Adequate communication skills. Some evidence of teamwork. Reasonable cultural fit. - **2 (Below Average)**: Weak communication skills or limited collaboration evidence. Questionable cultural fit. - **1 (Poor)**: Poor communication skills. No evidence of teamwork or collaboration.

Project Deliverable Evaluation Rubric

Parameter 1: Correctness (Prompt & Chaining) (Weight: 30%)

Description: Evaluate whether the solution correctly implements the required functionality, properly uses AI prompts and chain-of-thought reasoning, and

produces accurate results.

Scoring Guide: - **5 (Excellent)**: All requirements fully implemented and working flawlessly. AI prompts expertly crafted with clear reasoning chains. Edge cases handled properly. Output consistently accurate. - **4 (Good)**: Most requirements implemented correctly with minor issues. AI prompts well-structured with good reasoning. Handles common cases effectively. - **3 (Satisfactory)**: Core requirements implemented but with some issues. AI prompts work but could be improved. Some edge cases not handled. - **2 (Below Average)**: Partial implementation with significant functionality missing. AI prompts poorly structured. Many edge cases fail. - **1 (Poor)**: Minimal functionality implemented. AI prompts ineffective or missing. Core requirements not met.

Parameter 2: Code Quality & Structure (Weight: 25%)

Description: Assess code organization, readability, adherence to best practices, proper use of TypeScript/NestJS patterns, and overall software architecture.

Scoring Guide: - **5 (Excellent)**: Exceptional code quality with clear architecture. Follows all best practices. Well-organized modules and services. Professional-grade structure. - **4 (Good)**: Good code quality with solid structure. Follows most best practices. Code is readable and well-organized. - **3 (Satisfactory)**: Acceptable code quality. Basic structure in place but inconsistent. Code is understandable but could be cleaner. - **2 (Below Average)**: Poor code quality with weak structure. Many best practices ignored. Code is hard to follow. - **1 (Poor)**: Very poor code quality. No clear structure. Code is messy and difficult to understand.

Parameter 3: Resilience & Error Handling (Weight: 20%)

Description: Evaluate how well the system handles errors, validates inputs, manages failures gracefully, and provides meaningful error messages.

Scoring Guide: - **5 (Excellent)**: Comprehensive error handling throughout. All inputs validated. Graceful failure recovery with meaningful error messages. Handles all edge cases elegantly. - **4 (Good)**: Good error handling in most areas. Input validation in place. Errors caught and handled appropriately. - **3 (Satisfactory)**: Basic error handling implemented. Some input validation. Errors caught but handling could be improved. - **2 (Below Average)**: Minimal error handling. Little to no input validation. Many edge cases not handled. - **1 (Poor)**: No error handling. No input validation. Application crashes frequently.

Parameter 4: Documentation & Explanation (Weight: 15%)

Description: Assess the quality of code comments, README documentation, API documentation, and overall explanation of design decisions.

Scoring Guide: - **5 (Excellent)**: Comprehensive documentation including detailed README, API docs, and design decisions. Code well-commented where needed. - **4 (Good)**: Good documentation with clear README and setup instructions. Key design decisions explained. - **3 (Satisfactory)**: Basic documentation present. README exists with setup instructions. Enough to understand and run the project. - **2 (Below Average)**: Minimal documentation. README incomplete or unclear. Difficult to understand or set up. - **1 (Poor)**: No meaningful documentation. Very difficult to understand the project.

Parameter 5: Creativity / Bonus (Weight: 10%)

Description: Reward innovative approaches, additional features beyond requirements, clever optimizations, or exceptional implementation quality.

Scoring Guide: - **5 (Excellent)**: Highly innovative solution with creative approaches. Multiple value-added features beyond requirements. Demonstrates deep understanding. - **4 (Good)**: Some creative elements or additional features. Good optimizations. Shows initiative beyond basic requirements. - **3 (Satisfactory)**: Meets requirements without significant creativity. Standard implementation approach. - **2 (Below Average)**: Very basic implementation. Barely meets minimum requirements. - **1 (Poor)**: Incomplete basic implementation. Below minimum expectations.