1. Title

AI CV Evaluator and Project Reviewer System

2. Candidate Information

Full Name: Rachmat Hidayat
Email Address: rachmat@example.com

3. Repository Link

GitHub Repository: github.com/rachmat-hidayat/ai-cv-evaluator
■ Important: Repository name avoids the word 'Rakamin' to minimize plagiarism risk.

4. Approach & Design (Main Section)

Initial Plan:
I began by analyzing the requirements for building an AI-assisted CV and project report evaluator. The project was broken down into three parts: backend API, LLM integration, and evaluation logic. Assumptions included that input data would be structured and JSON-based.

System & Database Design:
The backend was built with Express.js. There are two main endpoints: /evaluate for submitting CVs and reports, and /result/:id for retrieving the evaluation result. MongoDB was used to store job requests and evaluation results.

LLM Integration:
Ragie AI and Groq Cloud were used for retrieval and inference. Ragie handles document chunking and embedding, while Groq Cloud performs the LLM-based evaluation. Prompt design was modular, with separate templates for CV and project review.

Prompting Strategy:
Prompts followed a strict JSON response format with two main fields: score and feedback. This ensured consistency and easy parsing.

Resilience & Error Handling:
The system includes retry logic for LLM API failures and timeouts. Randomness in model output is controlled using temperature settings.

Edge Cases Considered:
Tested invalid CV formats, missing data, and incomplete project reports. The evaluator handles these by giving a score of 0 with a feedback message.

5. Results & Reflection

Outcome:
The system successfully evaluates CVs and project reports based on job descriptions and scoring rubrics. It produces JSON responses with stable feedback.

Evaluation of Results:

Scores were consistent when prompts were standardized. Variations occurred with ambiguous inputs.

Future Improvements:
Integrate embeddings caching, add multi-language support, and improve context retrieval speed.

6. Screenshots of Real Responses

Screenshots demonstrate API responses for endpoints:
• /evaluate → returns job_id + status
• /result/:id → returns final evaluation (scores + feedback)

Example response:
{ "result": { "cv_match_rate": 85, "cv_feedback": "The candidate demonstrates strong alignment with the job requirements." } }

7. (Optional) Bonus Work

Added webhook-based job completion updates and a simple front-end dashboard to visualize evaluation results in real-time.