1. **Title**
   AI CV Evaluator Project Backend Engineer
2. **Candidate Information**
   - **Full Name:** Aisah Atik Fitriani
   - **Email Address:** aisyahafitriani05@gmail.com
3. **Repository Link**
   Github: https://github.com/Fitriaii/AI_CV_Evaluator.git
4. **Main Section**
   - **Initial Plan:**

     The project began with an analysis of the main requirements and features needed. The system was divided into several small modules, namely:

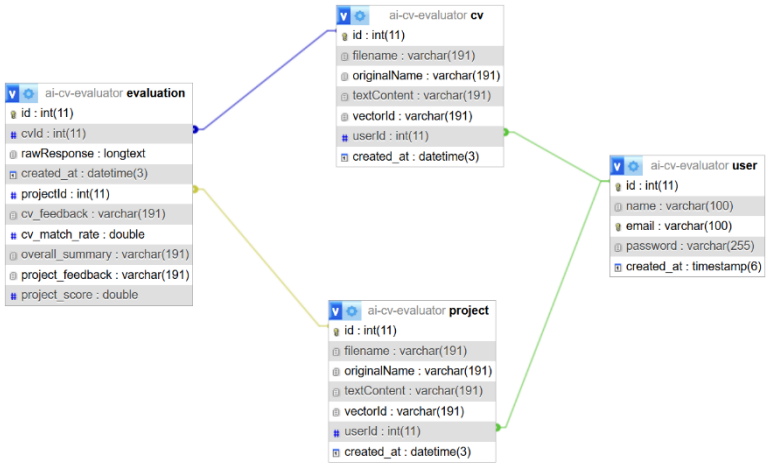
     - Authentication for user registration and login
     - Upload for uploading CVs and project reports
     - Extraction of file contents and embedding stored in the vectorDB, namely ChromaDB
     - Next, AI evaluation was performed using LLM to assess CVs and projects, and the results were stored in the database

     Assumptions:

     - The AI model is available and can be used for evaluation
     - Uploaded files are in PDF format
     - The backend system is capable of storing files and evaluation data

     Limitations:

     - Only creating the backend in project development
     - The AI only evaluates CVs and project reports
     - Using the free version of OpenRouter, so there are limitations during the embedding creation process
   - **System & Database Design:**
     - **API Endpoint Design**
       Designing backend endpoints to handle all key features, including:
       Authentication: /register, /login
       File Upload: /upload
       AI Evaluation: /evaluate
     - **Database Schema**



- **LLM Integration**
  The LLM used is OpenAI via OpenRouter because it is available for free, making integration easier despite usage limitations. The prompt is structured so that the AI provides consistent output in JSON format, including cv_match_rate, cv_feedback, project_score, project_feedback, and overall_summary. The RAG strategy is applied by taking embeddings from CV and project, searching for relevant reference documents in ChromaDB, then combining that context into the prompt before it is evaluated by the LLM. All processes are run sequentially to ensure accurate evaluation and the results can be directly saved to the database.
- **Prompting Strategy**
  const prompt = `
  You are an AI evaluator assessing a candidate's CV and Project Report.
  Reference Job Description:
  ${jobDesc}
  Reference Case Study:
  ${caseStudy}
  Candidate CV:
  ${cv.textContent}
  Candidate Project Report:
  ${project.textContent}
  Provide a detailed evaluation in strict JSON format as follows:
  {
  "cv_match_rate": number (0-1),
  "cv_feedback": string,
  "project_score": number (0-5),
  "project_feedback": string,
  "overall_summary": string
  }
  `;
- **Resilience & Error Handling**
  The system handles potential API failures, timeouts, or inconsistent results with a fallback mechanism. When generating embeddings or calling LLM, if an error occurs or the AI returns invalid data, the system provides embeddings or default values for evaluation so that the process continues to run. All API calls are wrapped in try-catch, so that the server remains stable and does not crash despite external disturbances or randomness from the AI.
- **Edge Cases Considered**
  One important scenario that was considered was the limitation of the free version of the OpenRouter API, which does not support embedding creation. To overcome this, the system uses pseudo-embedding as a substitute so that the evaluation process can continue even without the original embedding. Testing was conducted to ensure that the backend remained stable and that the AI continued to provide consistent evaluation results.

5. **Result & Reflection**
   - **Outcome**
     The backend implementation successfully handles CV and project report uploads, generates pseudo embeddings, and evaluates AI using OpenRouter in a stable manner. Evaluation results can be stored in the database and returned in a consistent JSON format. However, the limitations of the free version of OpenRouter prevent the original embedding process from running, so the evaluation quality may not be as optimal as when using official OpenAI embeddings.
   - **Evaluation of Result**
     The AI evaluation results are quite stable because the prompts are clear and the output is always in the same JSON format. However, as with the previous issue, because the free version of OpenRouter does not support native embedding, some evaluation scores may be inaccurate or slightly different. The use of pseudo-embedding helps the process to continue, but the quality of the assessment is not as accurate as when using native embedding.
   - **Future Improvements**
     With more time or access to more comprehensive tools, the system could use OpenAI's native embeddings to improve AI evaluation accuracy. In addition, a job queue could be added to handle heavy evaluation processes in the background to make the server more responsive. Currently, time constraints and the use of the free version of OpenRouter limit the creation of embeddings, so the solution must use less accurate pseudo-embeddings.