🎯 PROJECT OVERVIEW

This summary outlines the steps taken to build a user-friendly Interview Coach script that analyzes resumes against job descriptions, generates tailored interview questions, and evaluates practice answers—explained simply for everyone to understand. The script was developed to help job seekers prepare for roles in data science and machine learning, using AI for personalized career guidance. As of October 2025, I updated the models to current Groq offerings like **Llama-3.3-70b-versatile** for balanced performance.

🛠️ DEVELOPMENT STEPS

Step 1: Setting Up the Workspace

What I Did: Created a project folder named **resume-interview-assistant** to store all files, keeping the script's tools organized and separate from other programs on my computer.
Tools Used: Python and a virtual environment (though executed in Google Colab for simplicity).

Step 2: Installing the Main Software (Groq Library)

What I Did: Installed the Groq Python library (version 0.32.0 or later) in Google Colab using the !pip install groq command. Verified it worked by proceeding to imports without errors.
Tools Used: Google Colab and pip (for Groq library).

Step 3: Importing Libraries

What I Did: Imported essential Python libraries like json for handling API responses and Groq from the groq package to connect to AI models. This set up the foundation for sending prompts and parsing results.
Tools Used: Python (with json and groq libraries).

Step 4: Mounting Google Drive

What I Did: Connected Google Drive to the Colab environment using the drive.mount function, allowing secure access to stored files like the API key. The mount confirmed with a success message.
Tools Used: Google Colab and Google Drive.

Step 5: Loading the API Key

What I Did: Read the Groq API key from a text file (api-key.txt) in my Google Drive folder, stripped any extra spaces, and printed a confirmation message. This ensured secure authentication for all API calls.
Tools Used: Google Drive file reader and Python file handling.

Step 6: Prompt Engineering for AI Interactions

What I Did: Designed and refined detailed prompts for the Groq models to ensure accurate JSON outputs. For example, in analyze_resume_job_fit, I crafted a role-based prompt ("ACT as a professional career coach") specifying exact JSON structure for match scores, skills, and suggestions. Iterated on temperature settings (e.g., 0.3 for balance) and tested prompts for generate_interview_questions and evaluate_answer to include rationale, difficulty, and STAR method feedback. This step focused on making responses actionable and error-resilient.
Tools Used: Python string formatting, Groq API documentation, and iterative testing in Colab.

Step 7: Creating the InterviewCoach Class

What I Did: Defined a Python class called **InterviewCoach** to manage the script's core functions. It includes methods like:

- analyze_resume_job_fit (compares resume to job description with match scores and suggestions)
- generate_interview_questions (creates 3-5 targeted questions)
- evaluate_answer (scores user responses 0-10 with feedback)

Added error handling with fallback models (updated to **Llama-3.3-70b-versatile** and **Llama-3.1-8b-instant**) and demo data, plus model options for fast/balanced/quality performance.
Tools Used: Python (OOP with classes), Groq API, and JSON formatting.

Step 8: Initializing and Testing the Script

What I Did: Created an instance of the InterviewCoach class with the Groq client, listed available models, and ran end-to-end tests using sample data: a resume for "John Doe" (Senior Data Scientist) and a job description for "Senior Machine Learning Engineer."

Tests included:

- **Resume analysis:** Got 78% match score, listed skills (e.g., Python present, Docker missing), strengths/weaknesses, 3 improvements, 3 questions, and salary estimate ($85,000-$110,000)
- **Question generation:** Produced 3 mixed questions (e.g., on ML optimization)
- **Answer evaluation:** Scored a sample response 7/10 with strengths (e.g., "Relevant") and improvements (e.g., "Use STAR method")

All outputs printed successfully, handling any model errors with fallbacks.
Tools Used: Google Collab, Groq API models (Llama-3.1-8b-instant, Llama-3.3-70b-versatile), and sample text data.

Step 9: Saving Outputs for Proof

What I Did: Captured console outputs from the tests (e.g., match scores, questions, evaluations) as text logs or screenshots in a project folder. This documented the script's functionality, like "Match Score: 78%" and "Answer Score: 7/10."
Tools Used: Google Collab console/screenshots and a project folder.

✅ CONCLUSION

The Resume Interview Assistant is complete, successfully analyzing resumes, generating relevant questions, and providing constructive feedback. By setting up the workspace, installing Groq, engineering precise prompts for reliable AI outputs, building the class with robust fallbacks, and testing with real-world samples, I've created a practical tool for interview prep. The clear outputs and instructions ensure anyone can use it, demonstrating my skills in **AI integration, prompt engineering, Python development, and user-focused applications.**