

GenAI Virtual Interviewer – Proof of Concept (PoC)

ASSIGNMENT

CHAPTER

Prepared By:

Sheetal

M.Tech (Computer Science & Analytics)

NIT Delhi

Date:

April 6, 2025

Institution:

National Institute of Technology (NIT), Delhi

Executive Summary

In today's digital and remote-first world, **virtual interviews** have become essential in streamlining recruitment processes. Traditional interview systems often lack personalization, are time-consuming, and are prone to human bias. With the rapid evolution of Generative AI, there is an opportunity to reimagine how interviews are conducted — making them more **efficient**, **scalable**, and **objective**.

This Proof of Concept (POC), titled “**GenAI Virtual Interviewer**”, addresses these challenges by leveraging the power of **LLMs (Large Language Models)** to automate the end-to-end virtual interview process. The system is designed to:

- **Read and process resumes (PDFs)** to extract relevant skills and information.
- **Dynamically generate personalized interview questions** based on the candidate's resume and job role.
- **Evaluate candidate responses** using AI-based scoring mechanisms.
- **Generate detailed interview summaries and logs**, enabling easy review and comparison.

What This POC Demonstrates:

- **AI-driven question generation** tailored to individual resumes.
- **Resume parsing from unstructured PDFs** into structured data for context building.
- **Use of Retrieval-Augmented Generation (RAG)** for context-aware conversations.
- **LLM-based response evaluation** that mimics a human interviewer's feedback.
- **End-to-end workflow automation** from resume upload to interview summary.

Potential Impact:

- **Scalable recruitment** across multiple roles and industries.
- **Reduced interviewer workload**, enabling focus on high-value tasks.
- **Bias mitigation** through consistent AI-driven evaluations.
- **Enhanced candidate experience** with instant feedback and engaging conversations.

By combining document intelligence and conversational AI, this POC provides a compelling foundation for organizations aiming to **modernize and automate** their hiring pipelines using Generative AI technologies.

GitHub Repository: <https://github.com/sheetaI911/GenAI-Virtual-Interviewer-Proof-of-Concept-PoC->

Note: Certain PDF viewers may restrict hyperlink functionality. Manual access via copy-paste is recommended.

(Contains source code, sample resumes, and deployment steps, Sample output file (also attached on Page No. 14))

2. Problem Statement

2.1 AI-Driven Interviewing Challenges

Designing a truly effective AI-driven virtual interviewer goes far beyond simply asking static questions. It demands an intelligent system that can mimic human-like conversations while staying contextually relevant throughout the session. The primary challenges in this domain include:

- **Realistic, Human-like Dialogue**
A successful virtual interviewer must maintain natural and fluid conversation flow — adapting tone, question complexity, and phrasing based on the candidate's responses.
- **Context Retention Across Interactions**
Interviews are dynamic. The AI must remember prior answers and use that context to generate meaningful follow-up questions or adapt the direction of the interview.
- **Resume-Aware Question Generation**
Generic questions fail to provide deep insights. The system must tailor questions based on the specific content extracted from a candidate's resume — including skills, experience, education, and achievements.
- **Automated and Fair Evaluation**
Evaluating answers requires not only understanding what the candidate says but also assessing clarity, relevance, and alignment with job expectations — all while avoiding bias and maintaining consistency.

2.2 Resume PDF Challenges

Another core aspect of this POC is resume understanding — which is a non-trivial problem due to the nature of PDF formatting and the diversity of resume designs. Some key issues include:

- **Parsing Complexity**
Resumes may contain tables, columns, bullet points, charts, and inconsistent spacing — all of which make accurate text and data extraction difficult.
 - **Numerical & Tabular Data Recognition**
Information like years of experience, GPAs, and certifications is often embedded in varying numeric or tabular formats. Extracting this reliably is essential for personalizing the interview.
 - **Format Variability**
No two resumes are the same. The layout, font styles, sections, and language used can vary greatly. This makes it hard to define one-size-fits-all parsing rules.
 - **Precision in Information Extraction**
Inaccurate extraction can lead to irrelevant or misleading questions during the interview. Hence, a robust parsing mechanism is required to ensure the integrity and relevance of follow-up interactions.
- Together, these challenges highlight the complexity of building a truly intelligent and scalable virtual interviewer system that can simulate real-world hiring processes effectively.

3. Evaluation Metrics

To effectively evaluate the performance and impact of the GenAI-powered Virtual Interviewer, the following metrics are considered critical:

- **Response Quality**
 - Measures how well the AI generates meaningful, coherent, and job-relevant questions.
 - Evaluates grammar, tone, relevance to the role, and logical flow of the interview.
- **Context Awareness**
 - Checks if the system maintains memory of the candidate's previous answers.
 - Assesses how effectively it uses this context to generate intelligent follow-up questions.
- **Latency**
 - Tracks the time taken by the AI to process inputs and generate responses.
 - Ensures that the conversation feels natural and doesn't lag due to delayed outputs.
- **Bias Handling**
 - Evaluates if the AI remains fair, inclusive, and unbiased across different candidate backgrounds.
 - Detects gender, race, or experience-related bias in question generation or evaluation.
- **Resume Processing Accuracy**
 - Measures the correctness of data extracted from resumes, including text, tables, and numerical information.
 - Directly impacts how personalized and relevant the generated questions are.
- **Robustness**
 - Tests the system's ability to process resumes of various formats — including noisy, scanned, or poorly structured PDFs.
 - Evaluates the fallback mechanism in case of unreadable or incomplete data.

4. Model / Chain Architecture

This section outlines how the GenAI Virtual Interviewer processes input, generates dynamic questions, evaluates responses, and produces a summary—all in a seamless pipeline built on modern LLM capabilities.

4.1 LLM Choice

- **Model Used:** OpenAI GPT-3.5 Turbo / GPT-4 (can be swapped with any compatible LLM)
- **Reasons for Choosing GPT:**
 - **Zero-shot and Few-shot Learning:** Capable of generating contextual questions even with minimal training.
 - **API-Friendly:** Easily integrated with Python-based backend using OpenAI's robust API support.
 - **Natural Language Understanding:** Demonstrates high fluency and comprehension for resume-based Q&A.
 - **Memory & Context:** Efficiently holds multi-turn conversation logic and interview context.

4.2 Architecture Chain

A breakdown of how data flows through the system:

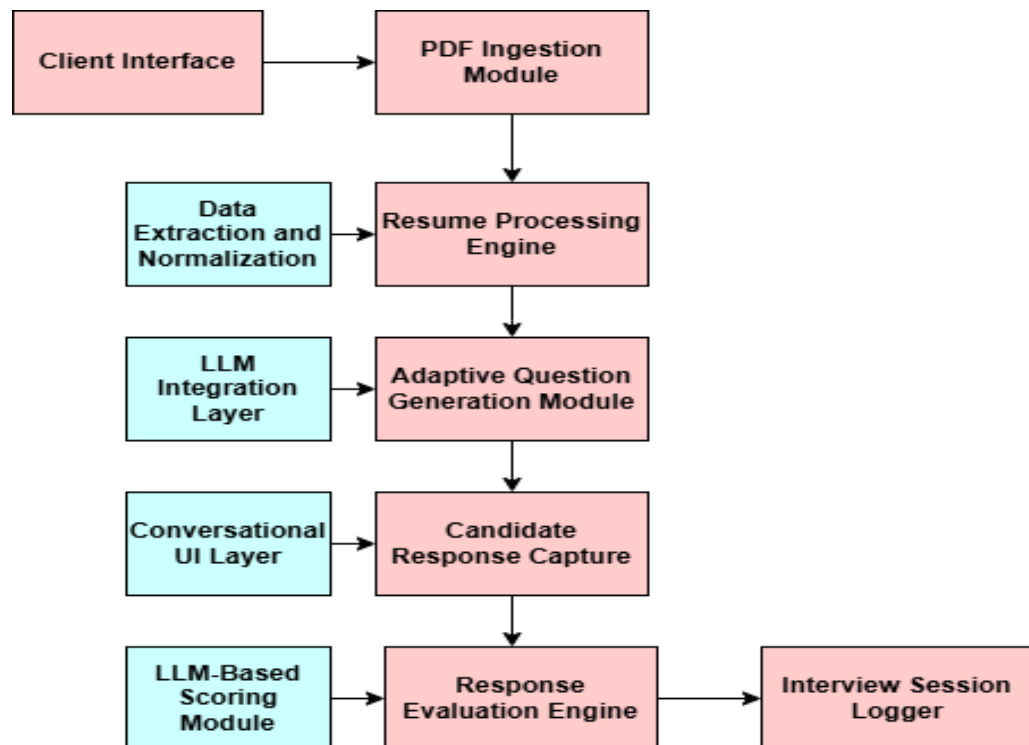


Figure 1. End-to-End Pipeline for AI-Driven Interview Generation and Evaluation

Detailed Steps:

1. PDF → Text Extractor

- Extracts raw text from resumes using PDF parsers like PyMuPDF, pdfminer, or pdfplumber.
- Cleans and preprocesses the text for further processing.

2. Embedding Generator + Vector DB

- Converts the resume text into semantic embeddings using sentence-transformers or OpenAI's text-embedding-ada-002.
- Stores embeddings in a vector database (e.g., FAISS or Chroma) for efficient similarity search.

3. RAG (Retrieval-Augmented Generation)

- Retrieves relevant parts of the resume based on context required to form intelligent questions.
- Provides the LLM with targeted snippets to avoid hallucination and improve specificity.

4. LLM-Based Question Generation

- Dynamically generates interview questions tailored to the candidate's resume and job role.
- Includes both general and technical questions, adapting as the conversation evolves.

5. Candidate's Response

- Candidate types (or optionally speaks) their answer in response to each generated question.

6. LLM-Based Evaluation

- The same or another LLM instance evaluates the candidate's answers.
- Uses predefined rubrics or natural prompts to assign scores and feedback (e.g., relevance, correctness, confidence).

7. Interview Summary Generation

- After the interview, the system generates a comprehensive summary in a PDF or JSON log.
- Includes key answers, evaluation scores, and insights like strengths or areas of improvement.

5. Document Processing Approaches

To ensure accurate and meaningful extraction of candidate information from resumes in PDF format, the project implements a robust document processing pipeline. This section outlines the tools currently used and proposed future enhancements for improved layout-aware parsing.

5.1 Tools Used in Current POC

The current Proof of Concept leverages the following Python libraries for PDF document parsing:

pdfplumber

- Used for extracting **raw text** from structured sections of resumes.
- Particularly effective in preserving line breaks and spacing for cleaner extraction.

PyMuPDF (fitz)

- Offers low-level control for reading PDFs, including **bounding boxes** and page layouts.
- Used for both **text extraction** and identifying **metadata**, which can assist in future layout-aware processing.

tabula-py (Optional for tabular resumes)

- Capable of extracting **structured tables**, if resumes include academic records or skill matrices in tabular form.

re (Regular Expressions)

- Applied for **numeric data extraction**, such as percentages, years of experience, graduation dates, or GPA.

5.2 What's Extracted

- **Raw Text:** Full textual content from all pages of the resume.
- **Tables (if present):** Parsed from PDFs using bounding boxes and detected layout elements.
- **Numerical Entities:** Identified using regex-based pattern matching (e.g., CGPA, dates, scores).

5.3 Future Enhancement Plans

To address limitations in parsing inconsistently structured or graphically complex resumes, we plan to explore advanced layout-aware document parsing solutions:

Layout-Aware Models

- **Donut (Document Understanding Transformer):** For visual-text parsing of scanned or layout-rich PDFs.
- **LayoutLMv3:** Combines textual, visual, and layout information to understand document structure effectively.

Cloud-Based Scalable Tools

- **Azure Form Recognizer:** For high-accuracy extraction of tables, key-value pairs, and text from diverse PDF layouts.
- **AWS Textract:** Offers robust OCR capabilities with strong layout understanding for documents with complex structure.

These additions will improve consistency, accuracy, and scalability in document processing, especially when onboarding resumes at scale or from diverse sources.

6. Tech Stack Documentation

Component	Technology	Why Chosen?
Backend	Python, Flask	Lightweight framework; easy integration with ML pipelines and REST APIs
Frontend	Streamlit	Ideal for building quick, interactive UI for demos and internal POCs
Resume Parsing	pdfplumber, PyMuPDF	Efficient for extracting structured/unstructured text and understanding layout
LLM Integration	OpenAI GPT API	Provides cutting-edge capabilities for Q&A, summarization, and evaluation
Vector Store	FAISS / In-memory store	For efficient retrieval of context during RAG-based question generation
Storage	JSON, FPDF	Lightweight format for storing results and generating structured summaries
Hosting	Google Colab / Vercel	Free-tier and quick deployment for demoing proof-of-concept functionality

7. System Architecture Diagram

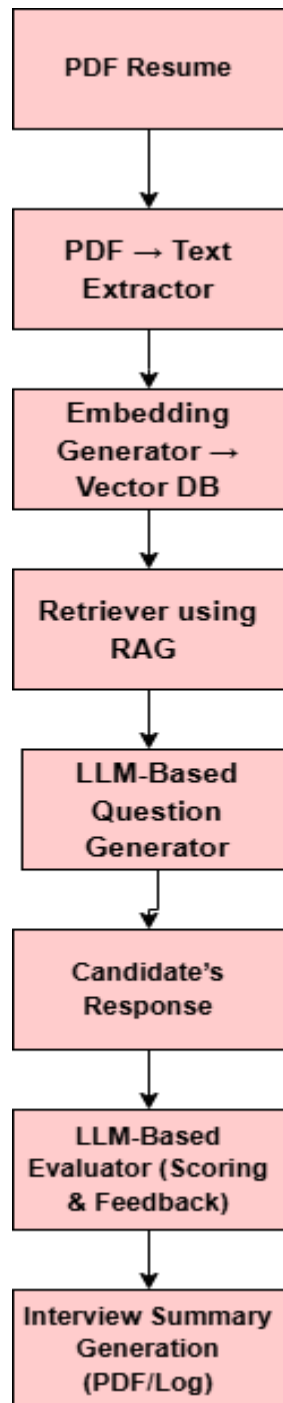


Figure 2: GenAI-Powered Virtual Interview Pipeline

8. Component Specifications

This section outlines the core components involved in the GenAI Virtual Interviewer system. Each module performs a distinct role, contributing to a seamless, intelligent interview experience.

1. Resume Reader

- **Purpose:** Extracts clean, structured text from uploaded PDF resumes.
- **Technology Used:** pdfplumber, PyMuPDF
- **Input:** PDF file
- **Output:** Extracted plain text (string or JSON format)
- **File Structure:** Stored in a temporary workspace or JSON structure for downstream processing

2. RAG Context Generator

- **Purpose:** Converts resume text into vector embeddings and retrieves context using Retrieval-Augmented Generation (RAG).
- **Technology Used:** sentence-transformers, FAISS or in-memory vector stores
- **Input:** Resume text
- **Output:** Relevant context chunks (text segments)
- **File Format:** Embeddings in .pkl or vector DB format (e.g., FAISS index)

3. LLM Question Generator

- **Purpose:** Generates personalized interview questions using the context retrieved from the resume.
- **Technology Used:** OpenAI GPT API or any LLM with prompt-engineering
- **Input:** Contextual resume data + system prompt
- **Output:** Natural language question(s)
- **File Format:** JSON (e.g., {"question": "Tell me about your experience with Python projects."})

4. Answer Evaluator

- **Purpose:** Scores the candidate's answer using another prompt to an LLM, evaluating on relevance, clarity, and depth.
- **Technology Used:** OpenAI GPT API (different role-based prompt)
- **Input:** Question + Candidate Answer
- **Output:** Score + Explanation (JSON)

Format:

```
{  
  "score": 8.5,  
  "feedback": "Good explanation, but could include more technical details."  
}
```

5. Summary Creator

- **Purpose:** Combines questions, answers, and scores to generate a final interview report.
- **Technology Used:** FPDF, reportlab, or json-to-pdf logic

- **Input:** JSON log of Q&A + scores
- **Output:** PDF/Log File (final summary)
- **File Format:** summary.pdf, session_log.json

9. GenAI Strategy

This section outlines the strategic approach behind leveraging Generative AI (GenAI) in building the virtual interviewer, focusing on prompt design, evaluation logic, and future enhancements.

Prompt Engineering

- **Question Generation Prompts:**
 - Designed using **few-shot prompting**, where the LLM is shown examples of how to generate role-specific, resume-based questions.
 - The prompt includes extracted **contextual chunks** from the candidate's resume to make the questions personalized.
 - Example prompt:
“Given the following resume context, generate an interview question for a Data Scientist role. Focus on technical depth and real-world application.”
- **Context Awareness:**
 - The system leverages **RAG (Retrieval Augmented Generation)** to feed the most relevant resume snippets to the LLM, ensuring **contextual and coherent follow-up** questions.
 - A **session state** is maintained to store previously asked questions and answers for continuity.

Evaluation & Judging Prompts

- **Rubric-Based Evaluation:**
 - A second LLM prompt is used to judge candidate answers on metrics such as:
 - Relevance to the question
 - Technical depth
 - Clarity and articulation
 - The evaluation prompt is designed as a **scoring rubric** (e.g., scale of 1–10 with reasoning).
 - Example:
“Evaluate the following response for clarity, correctness, and depth. Assign a score and provide brief feedback.”

Embedding Strategy

- Resume content is split into meaningful chunks and embedded using a **sentence transformer** model.
- These embeddings are stored in an in-memory or FAISS vector database.
- On question generation, the system **retrieves top-K relevant chunks** using cosine similarity to guide the LLM.
- This improves both **speed** and **accuracy** of contextual responses.

Future Enhancements


- **Voice Input Support:** Integrate STT (Speech-to-Text) modules like Google Speech API for spoken answers.
- **Facial Expression Analysis:** Integrate CV models (OpenCV + DeepFace) to analyze candidate behavior during the interview.
- **Multimodal Understanding:** Add capabilities to process visual resumes or attachments alongside textual data.
- **Fine-Tuning:** Train or fine-tune a custom interview-oriented LLM using anonymized logs for improved domain adaptation.
- **Session Persistence:** Use Redis or in-memory DBs to maintain state across interviews in multi-turn formats.

10. Scalability and Performance Plan

Area	Strategy
Scaling	Docker containers for microservices
Load Handling	Queue systems (e.g., Celery)
Caching	Store vector DB / resume info temporarily
Speed	Minimal API calls, reuse embeddings
Deployment	Use GitHub + Vercel / Colab for PoC phase

11. Sample Output

Deploy





GenAI Virtual Interviewer

Upload your resume (PDF format)

 Drag and drop file here
Limit 200MB per file • PDF

Browse files

 Sheetal_resume (3).pdf 203.1KB ×



Extracted Resume Content

SHEETAL

Master of Technology

232211030@nitdelhi.ac.in

in Computer Science and Analytics

sheetal.delhi.19@gmail.com

National Institute of Technology, Delhi

+91-9084338064


L

Deploy

Extra-Curricular Activities

- Session Coordinator, 12th International Conference on Big Data Analytics in Astronomy, Science & Engineering organized by NIT Delhi, University of Aizu, Japan & IIT Delhi. 2024
- GATE Qualified, Graduate Aptitude Test in Engineering. 2023, 2025

Generate Interview Question



Interview Question

Can you explain how your previous role prepared you for this position?

Your Answer:

In my most recent role as a content writer, I was able to develop an advanced skill set in content manag

Submit Answer

LLM Score: 8/10