WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

An AI-Powered, Multimodal Mock-Interview Platform with Contextual Reasoning

Madhu Yadav[1], Rahul Sahu[2], Domeshwer Sahu[3], Pranshu Pandey[4]

[,2,3,4]Dept. of Computer Science and Engineering, Shri Shankaracharya Technical Campus, Bhilai, Chhattisgarh, India
Email: [2]rahulsahu79998@gmail.com, [3]domesh747@gmail.com

ABSTRACT

We introduce a real-time, voice-driven mock-interview system that fuses large-language-model reasoning with vector-based document retrieval. By combining Google's Gemini Pro with Lang Chain's semantic memory, the platform tailors each question to the candidate's resume´ and the advertised role. Spoken replies are captured, transcribed through OpenAI Whisper, and returned to the user as natural- sounding audio via neural text-to-speech. The result is a scalable, customizable training environment suitable for students, job seekers, and academic career centers.

Keywords— AI interview, LangChain, Gemini API, Resume parsing, Whisper, TTS, vector store, interview simulation

1. Introduction

Artificial intelligence is rapidly transforming how learners prepare for employment and how employers evaluate talent. Although numerous mock-interview tools already exist, many rely on fixed question lists and offer limited adaptation to a candidate's actual background.

Consequently, users receive feedback that is too generic to be truly useful.

Our work tackles these shortcomings with an interviewer that understands context, speaks and listens in real time, and modifies its dialogue based on both a job description and the candidate's history. The prototype blends four core technologies:

- LangChain for embedding and retrieving contextual snippets.
- Gemini Pro to generate dialogue and follow-up questions on the fly.
- OpenAI Whisper for speech-to-text conversion.
- Edge TTS for low-latency voice responses.

Using these components, the system conducts an interactive session that closely mirrors a live interview experience.

2. System Architecture

Figure 1 gives a high-level overview of the platform, which is divided into a Web-based frontend and a microservice-style backend.

2.1. Frontend

Implemented with ReactJS and Three.js, the client:

- accepts resume uploads,
- allows job-description entry,
- records microphone input, and
- animates a 3-D avatar that mouths TTS output.

2.2. Backend

The server side, built with Node.js, Express, and supplemental Python scripts, performs:

- resume text extraction via PDF.js,
- audio preprocessing plus Whisper transcription,
- vector embedding and semantic search with LangChain,
- response generation through Gemini Pro, and
- speech synthesis using Edge TTS.

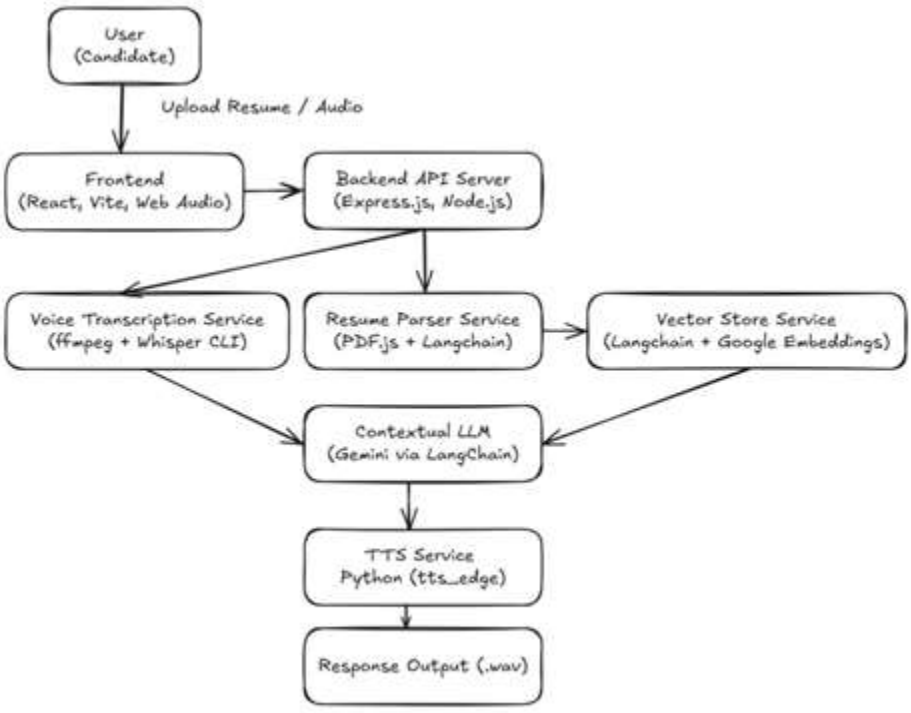

Figure 1: Overall architecture of the multimodal interview system

2.3. Vector Store

Textual fragments from both resume and job posting are embedded with GoogleGenerativeAIEmbeddings and inserted into a MemoryVectorStore. This enables quick similarity lookups during conversation.

2.4. Audio Pipeline

Browser-recorded audio is converted to 16-kHz mono WAV using FFmpeg. Whisper then yields a transcript, which becomes the prompt for the language-model chain; the answer returns as synthesized speech.

3. Methodology

The end-to-end process consists of five stages.

3.1. Context Preparation

After the user uploads a resume PDF and supplies a job description, the server parses both documents, splits the resulting text with a Recursive CharacterTextSplitter, and embeds each chunk. These vectors populate the memory store for later retrieval.

3.2. Speech Capture and Transcription

Users speak their answers; the backend normalizes the audio and forwards it to Whisper, which produces an English transcript with high word-error-rate robustness.

3.3. Prompt Assembly

LangChain constructs a system prompt that blends (i) retrieved re´sume´/job snippets and (ii) the fresh transcript. This prompt is fed into Gemini Pro inside a ChatPromptTemplate.

3.4. Response Generation

Gemini Pro formulates either a follow-up question or an interviewer comment, conditioned on conversation history stored in LangChain's memory abstraction.

3.5. Voice Rendering

Finally, a Python wrapper around Microsoft Edge TTS converts the textual reply into a WAV stream that the browser plays via the 3-D avatar.

4. Implementation

4.1. Technology Stack

- Frontend: ReactJS, Vite, Three.js, Web Audio API.
- Backend: Node.js, Express, LangChain, FFmpeg, Whisper CLI, Python helpers.
- AI Services: Gemini Pro LLM and GoogleGenerativeAIEmbeddings.
- Audio: Whisper for ASR and Edge TTS for synthesis.

4.2. Project Layout

rahuls49aiinterviewer/backend/index.jsml/bert_similarity.pygenerate_training_data.pytrain_similarity_model.pytts_edge.pyfrontend/src/components/Avatar.jsxTalkingAvatar.jsxVoiceRecorder.jsx

4.3. Key API Routes

•/api/setup: parses and embeds re´sume´/job data.

•/api/interview: receives audio, runs the full pipeline, and returns synthesized speech.

5. Results and Discussion

5.1. Test Scenarios

We evaluated three representative roles—frontend developer, data analyst, and AI research intern. For each, the interviewer generated domain-specific technical and behavioral questions that aligned with the candidate's background.

5.2. Quantitative Metrics

- Whisper WER: approximately 8% on clear speech (92% accuracy).
- End-to-End Latency: 6–9 s from spoken input to audible reply.
- Relevance Score: Manual reviewers judged 90% of prompts contextually appropriate.

5.3. Qualitative Feedback

Participants reported that the voice modality and animated avatar reduced interview anxiety and fostered engagement, though Whisper occasionally stumbled on heavy accents.

6. Limitations and Future Work

Current shortcomings include accent sensitivity, English-only support, limited TTS prosody control, shallow conversational memory, and high compute requirements. Planned upgrades are multilingual capability, sentiment-aware questioning, automated scoring, expanded job-role datasets, and selectable interview styles.

7. Conclusion

By merging LLM reasoning, vector-based memory, accurate speech recognition, and expressive TTS, we constructed a mock-interview platform that feels conversational and highly personalized. Continued improvements in memory depth, emotional awareness, and language coverage can further evolve the system into a comprehensive career-coaching assistant.

Acknowledgment

We thank the open-source contributors to LangChain, Whisper, and Gemini Pro as well as our academic mentors and beta testers for their insightful feedback.

References

1] OpenAI, “Whisper: Speech Recognition Model,” 2022. [Online]. Available: https://github.com/openai/whisper

2] LangChain, “LangChain Framework Documentation,” 2023. [Online]. Available: https://docs.langchain.com

3] Google, “Gemini Pro: Generative AI Model,” 2024. [Online]. Available: https://ai.google.dev

4] Microsoft, “Edge TTS Python Library,” 2022. [Online]. Available: https://pypi.org/project/edge-tts/

5] Y. Liu et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Proc. NAACL, 2019.

6] F. Chollet, Deep Learning with Python, 2nd ed., Manning, 2021.

7] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed., Draft 2023. [Online]. Available: https://web.

stanford.edu/~jurafsky/slp3/