WWW.IJRPR.COM

International Journal of Research Publication and Reviews

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

AI Interview Assistant: an AI-Powered Platform for Mock Interviews, Answer Evaluation, Communication Analysis and Resume Assessment

Dr. Mrs. A. P. Rathkanthiwar [a], Rohan Virutkar [a*], Smital Burde [a], Makarand Bamhane [a]

[a] *Department of Electronics and Communication Engineering, Priyadarshini College of Engineering, Nagpur, Maharashtra, India*

A B S T R A C T

Preparing for a technical interview takes repeated practice and honest feedback, and both are hard to find for many students. This paper describes the AI Interview Assistant, a web application that brings mock interviews, automated answer feedback, spoken-answer processing, communication analysis, resume checking and progress tracking into one tool. The application is written in Python with a Streamlit interface. Language tasks are handled by a hosted large language model reached through an API, spoken answers are turned into text by a speech-recognition service, resume text is extracted from PDF files, and results can be kept in an SQLite database and reviewed on a dashboard. The system follows a modular pipeline: input collection, question generation, answer capture, transcription where needed, evaluation, score aggregation, storage and visualization. It is intended as a practice aid, not as a substitute for human interviewers or hiring decisions. No accuracy figures are claimed here; the paper instead sets out the architecture, the implementation approach, a protocol for testing against reference answers, human raters, word error rate and response time, and the limits of the current design.

Keywords: Mock interview, large language model, speech recognition, natural language processing, resume analysis, applicant tracking, performance evaluation, generative AI.

1. Introduction

Technical interviews test more than knowledge. They also reveal how clearly a candidate reasons, explains and responds under pressure. Engineering students and recent graduates usually need many practice rounds before they can do this comfortably, and they often cannot tell which parts of their preparation are weak.

The usual preparation routes are reading question banks, watching videos, rehearsing with friends and, when possible, sitting a mock interview with an experienced person. All of these help, but experienced interviewers are scarce, feedback depends on who happens to be available, and it is difficult to compare one practice session with the next.

Recent progress in language models and speech recognition makes a different approach practical. Large language models can produce context-aware questions and structured comments on an answer (OpenAI, 2023), and modern speech-recognition models can transcribe varied speech with reasonable robustness, although quality still changes with audio conditions, accent and language (Radford et al., 2023). Together they can support a practice tool that listens to or reads an answer and responds within seconds.

This paper presents such a tool. The AI Interview Assistant combines question generation, text and voice interviews, answer evaluation, communication indicators, resume-to-job-description comparison and a performance dashboard in a single interface. Instead of training a new model, the work integrates existing hosted services into a usable workflow; language-model inference runs on an external API, not on a model trained for this project.

The contributions of the paper are:

a modular architecture that joins interview practice, speech processing, resume assessment and progress tracking;

a description of the processing pipelines for text answers, voice answers and resume matching;

a clear separation between automated estimates and verified measures of ability; and

a reproducible plan for testing the system, covering transcription error, scoring agreement, resume matching and latency.

2. Problem definition and objectives

2.1 Problem statement

Interview preparation is usually spread across several unconnected tools, one for questions, another for recording answers, another for resumes. Candidates therefore lack a single place where they can practise, get feedback and see their progress over time, and the quality of the feedback they receive depends on reviewer availability. The problem addressed here is to design and build a modular AI-assisted platform that offers these functions together while being open about the difference between an automated estimate and a verified judgement of ability.

2..2 Objectives

Build an easy-to-use application for mock interview practice.

Generate interview questions suited to a chosen role and difficulty.

Accept both typed and spoken answers.

Evaluate answers with an AI-based assessment step and return structured feedback.

Compute communication-related indicators from the data that is available.

Extract resume text and compare it with a job description.

Store results and present them on a dashboard.

Document limitations and define a repeatable testing method.

2.3 Scope

The system is meant for practice, learning and self-assessment by engineering students, fresh graduates and other candidates. It does not decide whether anyone is employable. Scores are estimates produced by the implemented evaluation process and should be read alongside human judgement.

3. Related work

3.1 Large language models

Large language models generate and analyse text from instructions and context, and most current models build on the transformer architecture (Vaswani et al., 2017). The GPT-4 technical report describes one such model and its results on professional and academic benchmarks, while also noting its limitations (OpenAI, 2023). In an interview setting a model can propose questions, judge whether an answer addresses the question, point out missing points and suggest improvements. Its output can, however, contain mistakes or change between runs. Work on using language models as judges shows that they can agree well with human preferences in some settings but are affected by biases such as favouring longer answers (Zheng et al., 2023), so automated scores need to be checked against human ratings.

3.2 Automatic speech recognition

Speech recognition turns spoken language into a transcript. Whisper, trained on a very large collection of weakly supervised audio, is a widely used example (Radford et al., 2023). In an interview assistant the transcript sits between the audio and the evaluator, so transcription mistakes pass straight into the evaluation. A misheard technical term can change the meaning of an answer. For this reason transcription quality has to be measured separately from answer-evaluation quality.

3.3 Interview practice and assessment systems

Earlier research has looked at generating mock interview questions and dialogue. EZInterviewer, for instance, generates interview questions in a conversation and was assessed with human raters (Li et al., 2023). Other studies predict how well job-interview answers are explained from their text. These efforts show that generating fluent questions or comments is one problem and judging answers reliably is another. The same caution applies to the present system.

3.4 Resume analysis and job matching

Resume assessment normally means extracting text, finding skills and experience, and comparing them with a job description. Simple versions count matching keywords; more advanced ones use semantic representations or language models. These are different techniques with different strengths: keyword matching only tells whether chosen terms appear, whereas semantic methods try to relate differently worded statements. Text extraction from a

PDF is therefore not the same as understanding a resume, and a score produced by this module should not be presented as a score from any commercial applicant tracking system.

3.5 Research gap

The technologies above are useful building blocks, but combining them in one practice workflow still requires careful coordination of user interaction, generation, transcription, evaluation, storage and display. This paper contributes that integration. It does not introduce a new speech-recognition architecture or a newly trained language model, and it makes no claim of better evaluation accuracy; such claims would need separate experimental evidence.

4. Proposed system

4.1 Architecture

The application is organised in five layers (Fig. 1) so that each function can be maintained and tested on its own.

Presentation layer. A Streamlit interface (Streamlit Inc., n.d.) for setting up interviews, entering answers, uploading resumes and viewing results.

Application layer. Controls the workflow, validates inputs, calls the processing modules and prepares results for display or storage.

AI processing layer. Uses a hosted language model through an API for question generation, answer feedback and resume comparison.

Speech processing layer. Accepts audio and obtains a transcript from the configured speech-recognition service.

Data layer. Uses SQLite (Hipp, n.d.) to keep interview records and results where the storage option is enabled.

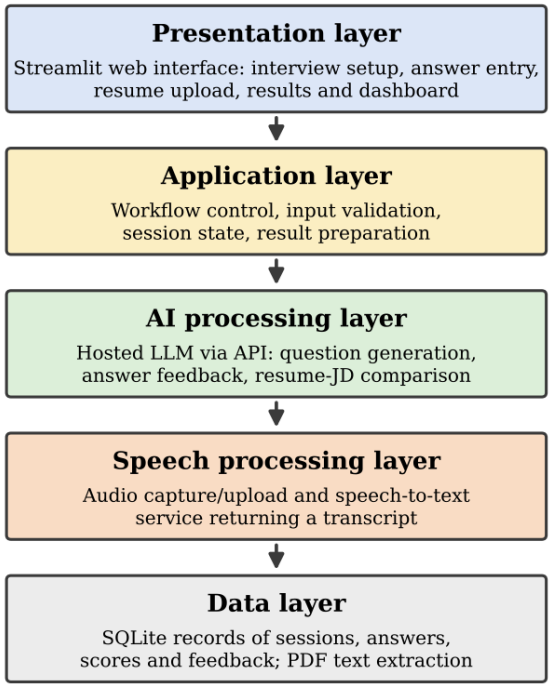


Fig. 1 - Layered architecture of the AI Interview Assistant (schematic).

4.2 Workflow

A session proceeds as follows (Fig. 2). The user sets the role, difficulty and number of questions; the system generates and validates the questions; the user answers by typing or speaking; spoken answers are transcribed; the evaluation module analyses the answer; communication indicators are computed where supported; the results are combined, shown to the user and, when storage is enabled, saved for later review.

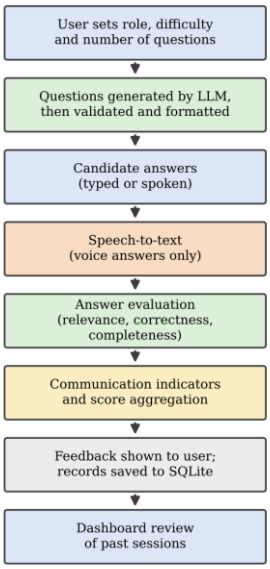


Fig. 2 - End-to-end workflow of one interview session (schematic).

5. Methodology

5.1 Interview configuration

Each session starts with user-chosen parameters, which may include the target role, difficulty, number of questions and resume information, depending on the interface version. These settings form the context sent to the question generator. Inputs are validated first so that invalid settings never reach later modules.

5.2 Question generation

The generator builds a prompt from the selected role and settings, sends it to the language-model API, receives the reply, and then checks and formats the questions before showing them. The quality of the questions depends on the prompt, the model, the context and the post-processing code, so generated questions should be reviewed for relevance, technical correctness, repetition and difficulty. An existing hosted model is used; none is trained for this project.

5.3 Text-based interview

In a text interview the candidate reads a question and types an answer. The answer, the question and the session context go to the evaluator, which returns feedback on relevance, technical correctness, completeness, strengths, weaknesses and suggested improvements, to the extent defined by the evaluation prompt. A dependable implementation also has to handle empty answers, API failures, malformed model output and missing fields in the returned evaluation.

5.4 Voice-based interview

In a voice interview the candidate speaks the answer. Audio is recorded or uploaded, sent to the speech-recognition service, and returned as text, which is then handled exactly like a typed answer (Fig. 3). In the reference configuration the transcription uses a Whisper-family model (Radford et al., 2023), Whisper Large v3, through the Groq API (Groq Inc., n.d.); the deployed configuration should be stated precisely when results are reported.

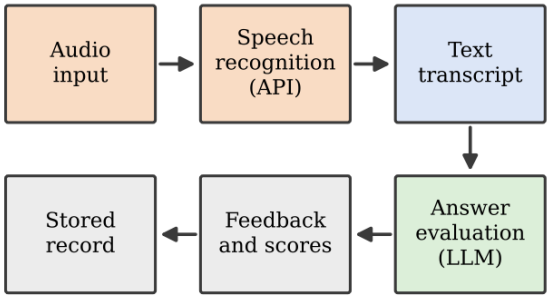


Fig. 3 - Voice answer pipeline: audio is transcribed and then evaluated like typed text (schematic).

Keeping transcription and evaluation as separate stages means errors can be traced to one stage or the other, and the speech service can be replaced without touching the evaluator.

5.5 Answer evaluation

The evaluator asks the language model to judge an answer against the question and return a structured result. Requesting a fixed output format, such as named fields with numeric scores and short comments, makes the reply easier to validate and store. The returned values are checked for range and completeness before use. Because model-generated scores can vary between runs and can be influenced by answer length or style (Zheng et al., 2023), they are treated as estimates. When several criterion scores are combined into an overall score, a weighted sum is used:

$$O = \Sigma\, w_k\, s_k\,, \quad \Sigma\, w_k = 1 \qquad (1)$$

where s_k is the score for criterion k and each w_k is a configurable weight. The weights are a design choice and have not been tuned against human judgements.

5.6 Communication analysis

Communication indicators are simple measures taken from the answer text or transcript where possible: answer length, sentence length, vocabulary variety, use of filler words and, when audio duration is available, speaking rate. They describe style, not the quality of reasoning, and they should be read with care. Speech-recognition systems often drop or smooth over hesitations, so filler counts taken from a transcript may understate what was actually said. Such indicators are prompts for reflection, not verdicts.

5.7 Resume and job-description analysis

The resume module extracts text from an uploaded PDF, cleans it, and compares it with the job description pasted by the user (Fig. 4). Depending on the implementation, the comparison may use keyword overlap, a language-model judgement, or both, and it returns missing skills and suggestions. The extracted text can be incomplete for scanned or heavily formatted resumes. The resulting match value is an indication of overlap with the supplied description and is not an applicant-tracking score from any employer system.

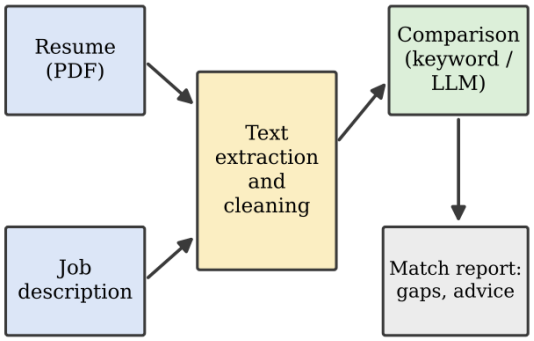


Fig. 4 - Resume assessment pipeline (schematic).

5.8 Storage and dashboard

When storage is enabled, session details, answers, scores and feedback are written to a local SQLite database (Fig. 5). The dashboard reads these records and shows summary figures, score trends and a history of sessions (Fig. 6), which lets a candidate see whether practice is producing improvement.

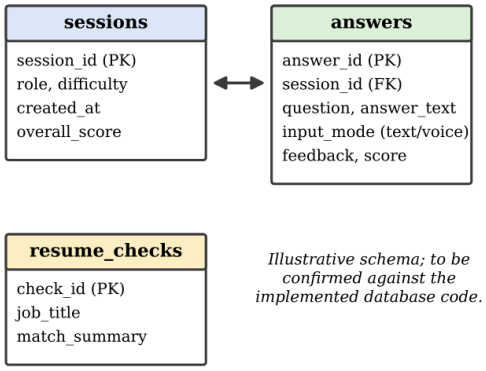


Fig. 5 - Illustrative data model for stored interview records.

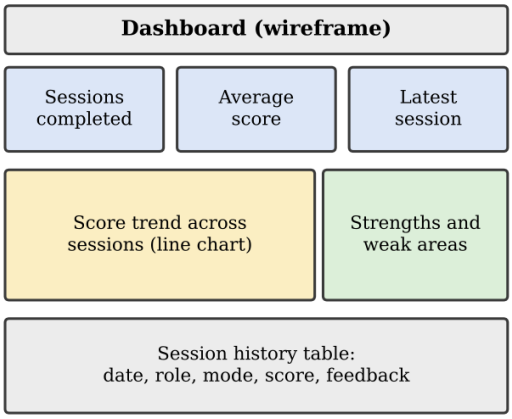


Fig. 6 - Layout of the performance dashboard (wireframe, not a screenshot of measured results).

6. Implementation

The application is implemented in Python. Table 1 lists the main components and their roles. Secrets such as API keys are kept outside the source code, for example in environment variables, and calls to external services are wrapped so that timeouts and errors produce a clear message instead of ending the session.

Table 1 - Main software components.

Component	Role
Python	Application logic and glue code
Streamlit (Streamlit Inc., n.d.)	Web interface and session state
Hosted LLM (API)	Questions, feedback, resume comparison
Speech-to-text service	Transcription of spoken answers
PDF text extraction	Resume text for analysis
SQLite (Hipp, n.d.)	Local storage of records and results

7. Evaluation methodology

No quantitative results are reported in this version. Effectiveness has to be shown by testing each component against a suitable reference (Fig. 7 and Table 2).

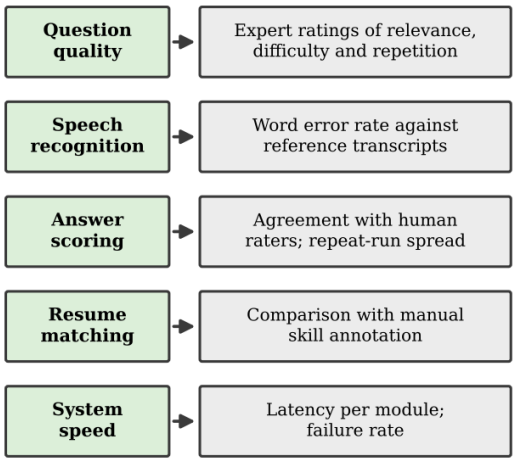


Fig. 7 - Planned evaluation of each component.

For speech recognition, word error rate is computed against reference transcripts (Morris, Maier, & Green, 2004):

$$WER = (S + D + I) / N \qquad (2)$$

where S, D and I are the numbers of substituted, deleted and inserted words and N is the number of words in the reference. Recordings should vary in accent, noise level and technical vocabulary. Answer scoring is compared with scores from human raters using correlation or weighted agreement, and the same answer is scored several times to measure run-to-run spread. Resume matching is compared with manual skill annotation, and response time is measured for each module.

Table 2 - Planned tests.

Component	Measure
Question generation	Expert rating of relevance, difficulty, repetition
Speech recognition	Word error rate on reference transcripts
Answer scoring	Agreement with human raters; repeat-run spread
Resume matching	Precision and recall against manual annotation
Latency	Mean and 95th-percentile time per module
Usability	User questionnaire after practice sessions

8. Applications

The platform can support students preparing for campus placements, graduates practising for first jobs, and training and placement cells that want an additional practice resource. Because sessions can be repeated at any time, it also suits self-paced learning where a human mentor is not always available.

9. Limitations and ethical considerations

Reliability. Language-model feedback can be wrong, inconsistent or biased, and scores are not validated against human experts in this version.

Transcription errors. Accents, noise and unusual terms can lower transcript quality and affect evaluation.

Shallow indicators. Communication measures describe surface features, not understanding or confidence.

Resume matching. Keyword overlap and PDF extraction miss meaning and can fail on complex layouts.

External services. The system depends on third-party APIs, so availability, cost and data handling follow those providers' terms.

Privacy. Answers, audio and resumes are personal data; they should be sent to outside services only with the user's knowledge and stored securely.

Intended use. The tool is for practice and must not be used to screen or rank candidates.

10. Conclusion and future work

This paper described the AI Interview Assistant, a modular application that joins interview-question generation, text and voice practice, answer evaluation, communication indicators, resume assessment and a performance dashboard. It relies on hosted language and speech services rather than a locally trained model, and it is positioned as a practice aid. The contribution is the integrated workflow and the testing plan, not a claim of high accuracy. Controlled experiments on transcription error, agreement with human raters, resume-matching quality and latency are the necessary next step before the quality of the feedback can be stated with confidence.

Planned work includes running the evaluation protocol described above, adding adaptive follow-up questions based on earlier answers, improving resume matching with semantic similarity, supporting more languages, adding rubric-based scoring that is calibrated against human raters, and studying how repeated practice affects candidate performance.

11. References

Groq Inc. (n.d.). *GroqCloud API documentation*. https://console.groq.com/docs

Hipp, R. D. (n.d.). *SQLite*. https://www.sqlite.org

Li, M., et al. (2023). EZInterviewer: To improve job interview performance with mock interview generator. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*. arXiv:2301.00972.

Morris, A. C., Maier, V., & Green, P. (2004). From WER and RIL to MER and WIL: Improved evaluation measures for connected speech recognition. In *Proceedings of Interspeech*.

OpenAI. (2023). GPT-4 technical report. *arXiv preprint* arXiv:2303.08774.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 28492–28518).

Streamlit Inc. (n.d.). *Streamlit documentation*. https://docs.streamlit.io

Vaswani, A., et al. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)* (pp. 5998–6008).

Zheng, L., et al. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*.