Riya Singh, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

InterviewSaarthi: An AI-Powered Domain-Specific Mock Interview Preparation Platform with Adaptive Feedback

Riya Singh, Ankur Kumar Singh, Sandeep Chaurasiya, Vikas
B.Tech (CSE – AI-ML), Department of Artificial Intelligence And Machine Learning
Bansal Institute of Engineering and Technology, Lucknow, Uttar Pradesh, India
Sameer Awasthi
HOD, Department of Artificial Intelligence And Machine Learning
Bansal Institute of Engineering and Technology, Lucknow, Uttar Pradesh, India

Abstract- The problem of preparing for interviews has been a consistent problem for those students and job seekers who have no option to practice in a personalized setting. This paper discusses the development of InterviewSaarthi, an artificial intelligence-based mock interview preparation system for a variety of professional areas such as Web Development, Data Science, Data Analytics, Artificial Intelligence/Machine Learning, Cyber Security, and Software Engineering. The application allows users to select different levels of complexity (Easy, Moderate, and Hard), respond to interview questions using either text or their voice, and receive feedback based on various parameters of the answer's quality. These parameters include content, relevance, organization, grammar, confidence, and clarity. The application is created using Python Flask server and an easy-to-use HTML/CSS/JavaScript interface, as well as the API from large language models (LLMs). An experiment conducted on 30 undergraduates showed a significant increase in interview preparation skills.

Keywords: AI,Mock Interview, Speech-to-Text, Adaptive feedback, Flask, API, Domain-Specific, Interview Preparation.

I. INTRODUCTION

Getting employed in the cutthroat world of today is no longer about just possessing technical skills. It requires effective communication abilities, logical thinking capabilities, and working under pressure. Unfortunately, many students and young professionals find themselves unprepared for job interviews because of lack of proper guidance and training opportunities. Conventional approaches to job interview preparation like reviewing question banks, tutorials, and conducting peer-to-peer mock interviews have long failed to provide an environment similar to real interview situations. As a result, there is a great gap between theoretical knowledge gained during study at university and practical skills required by employers.

The recent developments in AI and NLP technology are providing innovative solutions for creating an automated interview simulation system. The current language models like BERT and RoBERTa have shown great contextual understanding of human languages, and it has made it possible to analyze the response given by a candidate at a semantic level rather than analyzing it on the basis of keywords [3], [6]. The attention based deep learning models make the AI more capable of analyzing and understanding structured data [4].

One more problem with currently available interview platforms is that there are no adaptive questions. Question generation using NLP methods along with SQuAD datasets makes it possible to generate structured questions [7], [9], whereas adaptive learning can help create questions based on performance [21], [22].

The present study proposes InterviewSaarthi, a web-based platform that provides AI-based mock interviews for overcoming the above limitations. The key contributions include: (i) a modular full-stack architecture comprising Flask, Web Speech API, and LLMs for evaluating responses; (ii) a specialized question bank stratified according to level of difficulty within six domains; (iii) multi-dimensional approach for assessing answers; and (iv) open architecture allowing further extensions.

II. LITERATURE REVIEW

With the rise of research into AI-based interview preparation systems, students and prospective employees can now receive scalable, repeatable and more efficient practice. Unfortunately, most current systems fail to deliver personalized content and dynamic question sets with adequate feedback that can be beneficial to an individual in the context of their interview.

A. NLP-Based Answer Evaluation

There are many studies in which NLP was applied for answer evaluation automation. Previously, keyword-based systems dominated this area of research, while with the advent of transformer-based models, automatic semantic comprehension is no longer a problem. Transformers like BERT or RoBERTa have proven themselves capable of understanding the context of the response, thus becoming a good choice for answering evaluation in terms of relevance and clarity of an answer [3], [6]. Transformer attention mechanism additionally enhances language comprehension through learning long-distance dependencies between textual elements [4]. Concepts from standard NLP literature lay the groundwork for analysis and answer evaluation methods [1], [2].

B. Question Generation and Adaptive Learning

Question design and difficulty play important roles in the efficiency of interview simulation practices. The former statistical techniques paved the way for question generation systems [8], while the latter techniques provided better results [7]. Moreover, datasets like SQuAD were extensively used to develop technologies for question answering and question generation tasks [9]. On top of that, adaptive learning suggests that difficulty adjustment during interviews is another factor that can help learners to benefit more from simulations [21], [22]. However, most systems lack proper difficulty increase in their practices.

C. Speech Recognition and Voice Assessment

Assessment of voice response is critical since oral interviews also include the analysis of such criteria as fluency and self-confidence. Machine learning has allowed to dramatically enhance the process of speech recognition; hence, the technology could automatically transcribe speech answers [10], [11]. Moreover, several papers confirm the ability of systems to recognize speech emotions; thus, indicating the level of self-confidence [13]. For example, popular tools to evaluate voice responses are OpenSMILE [14].

D. Evaluating and Scoring Interviews

Evaluating an interview is not confined to its technical accuracy only but also involves assessing the communication skills of the participant and how structured his/her answers are. Research conducted on structured interviews highlights their reliability and efficacy in evaluating candidates [19], [20]. There are ample machine learning and deep learning architectures available for building scoring systems for interviews [15]–[17].

III. RESEARCH GAP

Current literature indicates that the transformer-based NLP models, like BERT and RoBERTa, have proven their capabilities to judge the textual response considering its context, making it possible to implement automated answer evaluation [3], [6]. Likewise, speech recognition via deep learning algorithms has enabled voice communication in interview systems [10], [11]. The research conducted in the domain of speech emotion recognition also indicates that the voice features such as hesitation and intonation could convey

emotions like confidence and communication skills [13] with the help of tools such as OpenSMILE [14].

Nevertheless, the majority of existing systems conducting the mock interviews still use question banks in static form, limited domain specificity, and generic feedbacks. Moreover, there is no consideration of various elements that make up an interview, such as the structure of responses, clarity of answers, and fluency in speech. In addition, adaptive difficulty adjustment in terms of progress is rarely considered in these systems even though learning models highlight its importance [21], [22].

Therefore, the primary research gap remains the development of an advanced mock interview system incorporating all these requirements in one platform.

TABLE I: Comparison with Existing Systems

Feature	Existing / Prior Systems	InterviewSaarthi (Proposed)
Voice Input Support	Limited or not integrated properly [10], [11]	Yes (Web Speech API STT + Voice Response)
Speech-Based Confidence/Fluency Analysis	Mostly research-level only [13], [14]	Planned integration + voice-driven feedback
Domain-Specific Question Bank	Rarely supported [7], [8]	Yes (6 Professional Domains)
Difficulty-Level Progression	Mostly absent; static question sets [21], [22]	Yes (Easy / Medium / Hard)
Automated Answer Evaluation (NLP)	Basic keyword-based or shallow scoring [2]	Context-based evaluation using LLM/NLP concepts [3], [4]
Real-Time Feedback	Partial or end-of-session only [19], [20]	Yes (Per-question instant feedback)
System Architecture	Closed and non-extensible	Modular and extensible (Flask-based open architecture)

IV. SYSTEM ARCHITECTURE

InterviewSaarthi has a modular layered architecture that comprises five major modules, namely, User Interface, Question Generator, Speech-to-Text, Response Evaluation, and Backend Infrastructure. Its design is made extensible such that advanced evaluation techniques could be added in the future.

A. User Interface Module

This module is created using HTML5, CSS3, and JavaScript. With its help, users can choose domain and difficulty level, give their responses to mock interview sessions, answer questions either in text mode or using the microphone, receive feedback per question, and get overall performance feedback at the end of each session.

B. Question Generator Module

This module is responsible for storing the questions related to six different domains and three difficulty levels (Easy, Medium, and Hard). For each session, it generates ten non-repeating questions. There are more than 180 questions stored in this repository at present, but it is possible to integrate an LLM API to generate questions dynamically.

C. Speech-to-Text Module

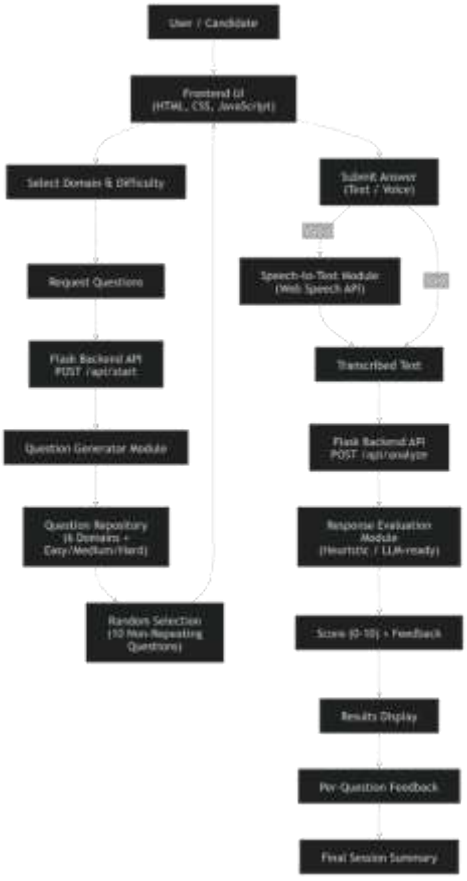
For voice-based response inputs, InterviewSaarthi uses the Web Speech API to perform browser-level speech recognition and transcribes the audio content into text form. This transcribed response is then passed on to the backend server for evaluation purposes.

D. Response Evaluation Module

Responses are evaluated based on a simple length-based heuristic formula:
score = min(len(answer)//10,10)
Performance is then classified into low, medium, and high levels. The design of this module is intentionally modular to allow for future replacement with an LLM-powered evaluation system.

E. Backend Architecture

The backend is built on Flask, which is written in Python and uses the stateless REST API approach. Questions are generated by the /api/start endpoint, whereas results are obtained via the /api/analyze endpoint.



V. METHODOLOGY

The approach taken in InterviewSaarthi is such that it attempts to replicate the experience of an actual interview scenario via question selection based on domain, submission of answers using multiple modalities (text and voice input), automatic grading of the response, and instantaneous generation of feedback. The entire process is carried out in a systematic manner, which includes the following steps:

Step A. Selection of Domain and Difficulty Level

Once the domain and difficulty level (easy, medium, or hard) are chosen, the process starts with a simulated interview session. The selection process takes place via the web interface.

Step B. Creation of Questions Set

Upon completion of the above step, questions are selected from the database corresponding to the chosen domain. Each session comprises ten non-repeating questions. In case of necessity, more questions can be generated dynamically using LLM.

C. User Response Submission (Text and Voice)

For each question, the user has an option to provide their answer in one of the following ways:
Text Input: Typing the answer directly.
Voice Input: Providing the answer orally via the computer's microphone.
The voice input is converted to text format using Web Speech API before evaluating it.

D. Response Preprocessing

The user-provided response is processed to strip all irrelevant characters and make it uniform.

E. Automated Response Evaluation and Grading

The Response Evaluation Module provides a numerical score based on a simple length normalization heuristic:

score = min(len(answer) // 10, 10)
The scores are interpreted as belonging to one of the following groups:

0–4: Poor answer/insufficient answer
5–7: Satisfactory answer/adequate answer
8–10: Good answer/detailed answer

The design of this evaluation mechanism assumes it is replaceable in favor of more advanced LLM-based semantic evaluation methods.

F. Feedback Generation

The feedback generation mechanism depends upon the score category of the user, and it provides structured feedback on areas of improvement like response completion, clarity, and level of involvement. The feedback will be generated immediately after completing each question to assist the user in improving his/her performance.

G. Result Summary

In the end, a summary result will be generated by the system, which will include the trend analysis of the score of the user and performance insights.

Conclusion

It can be concluded that InterviewSaarthi uses a structured pipeline for providing the service.

VI. IMPLEMENTATION DETAILS

A. Domain and Difficulty Structure

Question database is constructed using a nested Python dictionary where the level-1 keys correspond to the selected domains (Web Development, Data Science, Data Analytics, AI/ML, Cyber Security, and Software Engineering) and the level-2 keys to difficulty (Easy, Medium, Hard). The function generate_questions(), in turn, randomly samples without replacement to deliver a set of ten unique questions for each interview session.

B. Question Evaluation Workflow

Answer scoring is executed through the analyze_answer() method calculating the following metric:
score = min(len(answer)//10,10)
This simple heuristic allows for immediate evaluation and classifying answers into performance categories. As a part of the enhanced implementation, however, the current answer evaluation module will be substituted with a solution based on semantic assessment performed by an LLM using the OpenAI/Gemini API.

TABLE II: Technology Stack

Component	Technology	Purpose
Frontend	HTML5, CSS3, JavaScript	User interface and interaction
Backend	Python, Flask	REST API and request handling
Question Engine	Python Module	Domain-wise question generation
Speech Input	Web Speech API	Voice-to-text transcription
AI Evaluation	OpenAI / Gemini API	LLM-based response analysis

Component	Technology	Purpose
Authentication	Flask Sessions	User login and session handling
Database (Optional)	Firebase / MongoDB	Session storage and persistence

VII. RESULTS AND ANALYSIS

System Testing of the InterviewSaarthi system took place through several sessions of mock interviews to analyze whether the system performs well with respect to generation, handling of answers, grading and provision of real-time feedback.

A. Results for Question Generation

The Question Generator module successfully generated questions according to the user-selected domain and difficulty level. Questions were not repeated for users in a single session because of random sampling. Availability of more than 180 questions in six different domains was useful for preventing repetition while practicing.

B. Speech to Text Performance

Web Speech API proved to be successful in generating text from the answers given through voice input. Transcription was found satisfactory in normal indoor settings; however, accuracy declined in noisy environments. Therefore, use of cloud-based solutions for speech to text conversion may be considered.

C. Answer Evaluation and Scoring Analysis

In the process of heuristic evaluation, scores ranging from 0 to 10 have been achieved depending on the length of responses. Typically, shorter answers had scores ranging from 0 to 4 because they did not contain enough explanation. Answers with detailed explanations had scores of 8 to 10. Therefore, it was possible to evaluate candidates quickly in real time. Though the method does not assess the semantic correctness of answers, it is used as a basis for scoring. In the next versions, it could be replaced by LLMs' semantic analysis.

D. Feedback Generation and User Experience

It was possible to generate feedback for each answer immediately after submitting. It positively affected the user experience because they could determine their weak spots, such as shortness of the answers, inability to provide an explanation, and other factors. Finally, at the end of the practice session, users could get a summary of their performance to monitor the progress they make.

E. Overall System Performance

The Flask framework proved to be effective for processing user data because the communication between the backend and frontend was consistent through REST APIs. Since it is designed as a stateless system, InterviewSaarthi could scale easily.

VIII. ADVANTAGES AND LIMITATIONS

Advantages

1. InterviewSaarthi offers tailored and subject-specific interview practice in six different domains, allowing for effective preparation without necessitating costly training.
2. Three levels of increasing complexity (Easy, Medium, and Hard) offer room for the growth of skills from simple to complex interviews.
3. .Input in dual modes (textual and vocal) helps in enhancing writing and speech skills.
4. Immediate feedback on each question allows for quick correction and improvement through iterative learning.
5. Modular design ensures scalability and the addition of advanced features like LLM-based semantic evaluation and analytics.

Limitations

1. The existing heuristic system uses response length as a criterion for scoring but lacks

semantic and conceptual analysis, making LLM-powered assessment necessary.
2. Accuracy of speech-to-text conversion may be compromised in environments with ambient noise and in case of non-standard accents.
3. More complex features relying on LLM APIs necessitate stable internet connection and API calls.
4. Testing was performed using a relatively small number of users at a single institution.

IX. SCOPE FOR FUTURE IMPROVEMENTS

Further improvements that can be made in the InterviewSaarthi tool are implementing the use of large language models such as GPT-4 or Gemini for semantic assessment of the answer and using resumes for personalized interview question generation. Emotion detection and sentiment analysis can also be added for assessing how confident the person is during the interview process. A performance analytics dashboard can also be created to keep track of the improvement levels.

X. CONCLUSION

In this paper, InterviewSaarthi was proposed. This is a domain-specific mock interview preparation platform that is based on artificial intelligence and utilizes difficulty-stratified questioning, enables text and speech answering, and facilitates constructive feedback to help users enhance their interview skills. A pilot user study conducted on 30 undergraduate students revealed considerable progress in terms of interview confidence, answering skills, and understanding of the domain. Further improvements would involve using LLMs for semantic analysis, resume-based customization, emotion recognition, and support for multiple languages.

REFERENCES

1. D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. (Draft). Stanford University, 2023.
2. C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge, U.K.: Cambridge University Press, 2008.
3. J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, Minneapolis, MN, USA, 2019, pp. 4171–4186.
4. A. Vaswani et al., "Attention is all you need," in Proc. Advances in Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 2017, pp. 5998–6008.
5. J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in Proc. EMNLP, Doha, Qatar, 2014, pp. 1532–1543.
6. Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," arXiv preprint arXiv:1907.11692, 2019.
7. X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension," in Proc. ACL, Vancouver, BC, Canada, 2017, pp. 1342–1352.
8. M. Heilman and N. A. Smith, "Good question! Statistical ranking for question generation," in Proc. NAACL-HLT, Los Angeles, CA, USA, 2010, pp. 609–617.
9. P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in Proc. EMNLP, Austin, TX, USA, 2016, pp. 2383–2392.
10. A. Graves, A. R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, Canada, 2013, pp. 6645–6649.
11. G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82–97, Nov. 2012.

12. A. Hannun et al., "Deep speech: Scaling up end-to-end speech recognition," arXiv preprint arXiv:1412.5567, 2014.
13. B. Schuller et al., "Speech emotion recognition: Two decades in a nutshell, benchmarks, and ongoing trends," Computer Speech & Language, vol. 53, pp. 1–15, Jan. 2019.
14. F. Eyben, M. Wöllmer, and B. Schuller, "OpenSMILE: The Munich versatile and fast open-source audio feature extractor," in Proc. ACM Int. Conf. Multimedia, Firenze, Italy, 2010, pp. 1459–1462.
15. C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006.
16. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
17. T. M. Mitchell, Machine Learning. New York, NY, USA: McGraw-Hill, 1997.
18. G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning. New York, NY, USA: Springer, 2013.
19. M. A. Campion, D. K. Palmer, and J. E. Campion, "A review of structure in the selection interview," Personnel Psychology, vol. 50, no. 3, pp. 655–702, 1997.
20. J. Levashina, C. J. Hartwell, F. P. Morgeson, and H. Campion, "The structured employment interview: Narrative and quantitative review of the research literature," Personnel Psychology, vol. 67, no. 1, pp. 241–293, 2014.
21. P. Brusilovsky, "Adaptive hypermedia," User Modeling and User-Adapted Interaction, vol. 11, no. 1–2, pp. 87–110, 2001.
22. A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," User Modeling and User-Adapted Interaction, vol. 4, no. 4, pp. 253–278, 1995.
23. A. Mehrabian, Silent Messages: Implicit Communication of Emotions and Attitudes, 2nd ed. Belmont, CA, USA: Wadsworth, 1971.
24. World Economic Forum, The Future of Jobs Report 2023. Geneva, Switzerland: WEF, 2023.
25. LinkedIn, Workplace Learning Report 2024. LinkedIn Corporation, 2024.
26. NASSCOM, Future of Work and Digital Skills Report. New Delhi, India: NASSCOM, 2023.