Archives available at journals.mriindia.com

International Journal on Advanced Computer Theory and Engineering

ISSN: 2319 - 2526

Volume 15 Issue 01s, 2026

Leveraging NLP and LLMs for Intelligent Mock Interview Systems

[1]Prof. Sushil S. Deshpande, [2]Prof. Ujjwala S. Chaudhari
[1,2] Assistant Professor, IICT, MGM University, Chhatrapati Sambhaji Nagar
Email: [1]sdeshpande1@mgmu.ac.in, [2]uchaudhari@mgmu.ac.in,

Peer Review Information

Submission: 08 Dec 2025

Revision: 25 Dec 2025

Acceptance: 10 Jan 2026

Keywords

Machine Learning (ML), Natural language processing (NLP), large language modeling (LLM), Text-to-Speech (TTS)

Abstract

This paper presents the design and implementation of an **LLM and NLP-powered Mock Interview Application** that provides realistic, AI-driven interview simulations. The system integrates **Large Language Models (LLMs)** for dynamic question generation, contextual follow-ups, and personalized feedback, while **Natural Language Processing (NLP)** enables semantic understanding, sentiment analysis, and communication skill evaluation. **Text-to-Speech (TTS)** models facilitate voice-based interaction, and **Machine Learning (ML)** algorithms assess user responses based on content relevance, tone, and confidence. The modular architecture encompasses user authentication, interview setup, question generation, speech analysis, and performance visualization. Evaluation metrics include response accuracy, fluency, and emotional tone derived through NLP and LLM-based analysis. Future enhancements target **multimodal emotion recognition**, **eye-contact detection**, and **multilingual support**, positioning the platform as an advanced **AI-driven career readiness and communication training assistant**.

Introduction

In the contemporary digital era, Artificial Intelligence (AI) has become a cornerstone technology driving innovation across diverse sectors such as education, recruitment, and career development. Among its numerous applications, the use of AI to emulate human interactions and deliver adaptive, data-driven feedback has demonstrated significant potential in enhancing interview preparation and employability skills. The job interview remains a critical determinant in the recruitment process, assessing not only a candidate's technical expertise but also communication proficiency, confidence, and situational adaptability. Despite its importance, access to personalized interview coaching remains constrained by factors such as high cost, limited availability of expert mentors, and lack of scalability. Addressing these challenges necessitates the development of intelligent, automated systems capable of delivering personalized and accessible interview training. This topic presents an LLM and NLP-powered Mock Interview application designed to simulate realistic interview environments through advanced language and speech technologies. The system integrates Natural Language Processing (NLP), Large Language Models (LLMs), Speech-to-Text (STT), Text-to-Speech (TTS), & Machine Learning (ML) algorithms to enable dynamic question generation, semantic response analysis, and multidimensional feedback generation based on relevance, fluency, tone, and emotional indicators.

The platform supports domain-specific interview practice spanning technical, behavioral, and HR contexts, allowing users to enhance their readiness for real-world interview scenarios. The integration of these AI-driven components contributes toward scalable, accessible, and personalized career training, demonstrating the transformative potential of LLMs and NLP in professional skill development.

Liturature Review

In the modern recruitment ecosystem, mock interviews and skill assessments play a vital role in preparing candidates for real world job opportunities. Over the last few years, several systems and platforms have emerged to help students and professionals enhance their interview performance. However, the majority of these systems rely on static question banks, text-based evaluations, or human-led mock sessions, which often lack adaptability, personalization, and real-time analytical capabilities.

A-Traditional Manual Mock Interviews

Traditional mock interviews are usually conducted by experienced mentors or HR professionals in educational institutions or training centers. These sessions allow for personalized feedback and simulate real-world interactions. However, their scalability and consistency are major limitations. Each candidate receives subjective feedback, heavily influenced by the interviewer's perception. Additionally, manual interviews require significant human effort and scheduling, making them time-consuming, resource-intensive, and inaccessible to many learners. These systems also lack automated tracking of improvement over time or data-driven analysis of performance metrics such as confidence, tone, or vocabulary.

In summary, while traditional mock interviews provide human empathy and real-time dialogue, they are inefficient, costly, and inconsistent for large-scale training purposes.

B-Limitations Observed in Existing Systems.

1-Lack of Real-Time Interaction: Many systems are static and text-based, without natural two-way conversation using voice.

2-Limited Feedback Quality: Most systems provide numerical scores rather than descriptive, actionable feedback.

3- Data Privacy Concerns: Commercial systems often store user recordings without transparency.

4-Accessibility Issues: High-cost enterprise tools make AI interview preparation unaffordable for normal person.

5-Lack of Integrated AI Frameworks

Most existing works focus either on generating interview questions or analyzing candidate responses. Very few systems integrate both AI-driven question generation and automated response evaluation in a single application. This disjointed approach reduces realism and user engagement during practice sessions.

6-Limited Use of Voice and Speech Analysis

Many existing systems still rely on text-based input and output. This limits their capability to analyze vocal tone, pronunciation, hesitation, or confidence level — all of which are critical indicators of a candidate's communication skills. Real interviews are inherently speech-based; hence, systems that ignore voice data fail to capture essential behavioral cues.

7-Inadequate Emotional and Confidence Assessment

Few systems attempt to measure non-verbal or emotional aspects of communication. Emotion recognition and speech tone analysis can provide valuable insights into how confident, calm, or nervous a candidate sounds. This component remains largely unexplored in current academic systems and is often omitted in commercial tools due to complexity or cost.

8-Data Privacy and Ethical Limitations

Research-oriented solutions, on the other hand, often lack secure data handling mechanisms. Thus, a gap exists for developing an ethical, transparent, and secure AI interview platform that protects user data while ensuring fairness and Accountability in scoring.

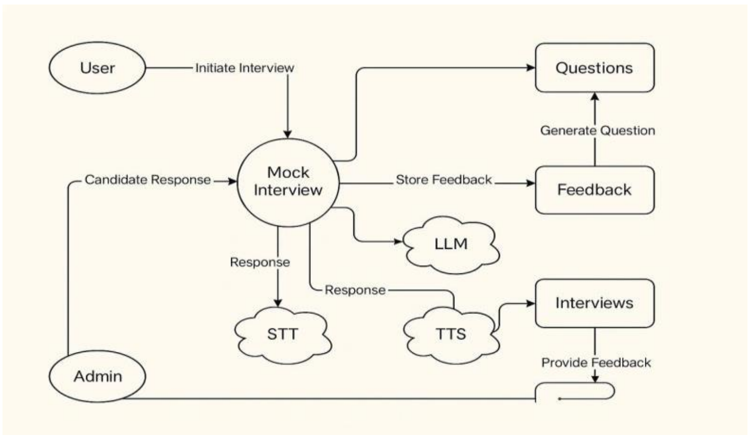

Figure I: Generalized proposed data flow model

Proposed Methodolgy

The proposed system implements an LLM and NLP-driven mock interview platform that leverages advanced artificial intelligence techniques to simulate real-world interview interactions and provide personalized feedback. The overall methodology comprises several integrated modules, as illustrated in Figure 1 (System Architecture), encompassing data processing, AI model interaction, user interface design, and performance evaluation.

1. System Overview

The system operates as an interactive application that enables users to participate in AI-driven mock interviews. Upon authentication, users select the interview domain (e.g., Technical, HR, Behavioral), and the system dynamically generates relevant questions using Large Language Models (LLMs).

2. Question Generation Module

This module utilizes LLM APIs (e.g., OpenAI GPT models) to generate contextually relevant and domain-specific interview questions. The LLM adapts its questioning pattern based on user responses, simulating a conversational flow similar to that of a human interviewer.

3. Speech and Text Processing Module

User responses can be provided via speech or text.

- Speech-to-Text (STT) models transcribe spoken answers into text for analysis.
- Text-to-Speech (TTS) models convert the AI interviewer's prompts into natural-sounding audio, enabling a voice-based interactive experience.

4. Response Evaluation Module

The transcribed or typed responses are analyzed using NLP and ML algorithms to assess:

- Content Relevance (semantic similarity to expected answers)
- Fluency and Coherence (grammatical and linguistic quality)
- Sentiment and Emotional Tone (confidence, positivity, engagement)

LLM-based sentiment and semantic analysis models quantify these parameters to generate a comprehensive feedback report.

5. Feedback Generation and Visualization

A structured feedback mechanism summarizes user performance metrics such as accuracy, communication skills, and confidence levels. The insights are visualized through charts and scores within the dashboard to support iterative self-improvement.

6. System Implementation

The front end is developed using ReactJS with Typescript, styled with Shadcn UI, and secured via Clerk Authentication. The backend integrates Firebase for real-time operations and Mongo DB Atlas for data persistence. API endpoints facilitate interaction between the web interface, AI services, and database layers.

7. Evaluation and Future Scope

System performance is evaluated using metrics such as response accuracy, fluency score, and sentiment alignment. Future improvements include emotion recognition, eye-contact detection, and multilingual interview support, advancing the platform toward a holistic AI-based career training ecosystem.

The diagram illustrates a feedback-driven loop where the **LLM** generates questions, **STT/TTS** enable interaction, and **NLP** modules evaluate responses to deliver personalized feedback for improved interview readiness.

1. **User Interaction**

The user initiates the interview session through the web application. The system begins by generating questions dynamically.

2. **Question Generation**

The Questions module interacts with the LLM (Large Language Model) to generate contextually relevant and domain-specific interview questions.

3. **Mock Interview Process**

The Mock Interview module serves as the central controller, handling communication between the user, AI components, and database. The user's candidate responses (either spoken or typed) are processed here.

4. **Speech Processing**

STT (Speech-to-Text) converts spoken responses into text for analysis.TTS (Text-to-Speech) enables the AI interviewer to communicate verbally, creating a natural interview experience.

5. **Response Evaluation and Feedback**

The LLM analyzes user responses for content relevance, tone, and fluency. The Feedback module stores and organizes evaluation results for user review.

6. **Data Management and Administration**

The Admin component manages interviews, monitors sessions, and provides aggregated feedback or performance analytics. Data from multiple interviews are stored and utilized to improve future feedback and evaluation.

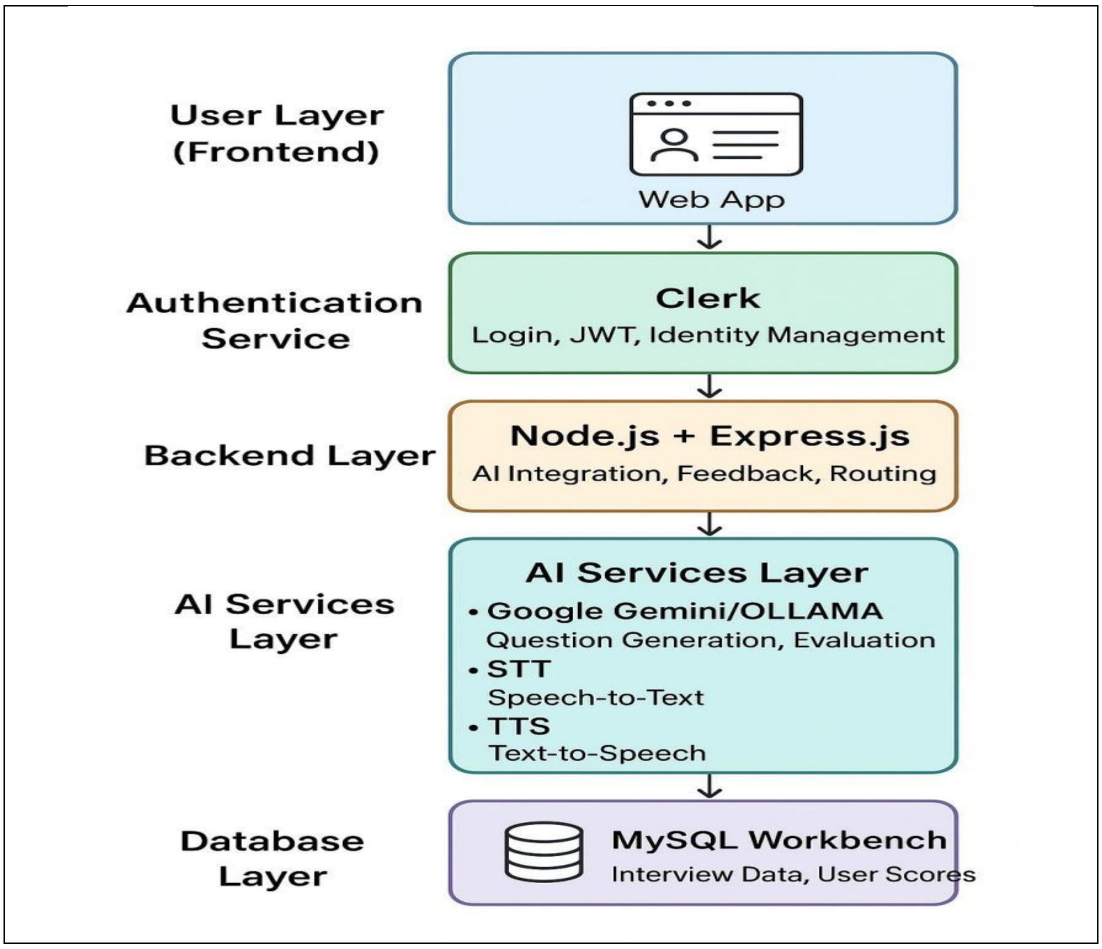

Figure 2: System Architecture

The system architecture for the AI-Powered Mock Interview Application follows a modular, cloud-ready, client–server design. The architecture separates responsibilities into frontend, backend, AI services, and databases to ensure scalability, maintainability, and security.

1. User Layer (Client):
Browser-based client built with ReactJS + TypeScript and Shadcn UI. Provides responsive UI, interview controls, and visual Feedback dashboards.
2. Frontend Layer:
Communicates with backend APIs, handles real-time audio capture/playback, displays TTS output, and renders evaluation charts.
3. Authentication Service:
JWT token manages secure login, session tokens (JWT), and user profile details.
4. Backend Layer:
Node.js + Express responsible for session logic, orchestrating AI calls (LLM, STT, TTS), scoring pipelines, and persisting Results to databases.
5. AI Services:
External or cloud-hosted services for Google Gemini / LLM (question generation and semantic evaluation), STT (speech-to- Text transcription), and TTS (interviewer voice).
6. Databases:
MySQL for interview/session storage and JWT token for real-time features, notifications, and optional analytics streams.
7. Third-party APIs:
Monitoring, logging, and CI/CD integrations for observability and maintainability.
8. Data Flow:
User → Frontend → Backend → AI Services → Databases → Feedback back to User.

Experimental Setup
To evaluate the performance and effectiveness of the proposed LLM and NLP-powered Mock Interview Application**,** a comprehensive experimental setup was designed. The setup includes the selection of appropriate technologies, system configuration, data collection methods, and evaluation metrics for assessing user performance and system capabilities. System analysis is a crucial stage in the Software Development Life Cycle (SDLC), as it involves understanding the system's requirements, feasibility, and technical

specifications.

1. Functional Requirements

Functional requirements specify the core operations and features of the system, ensuring that the application fulfills its intended purpose.

1.1 User Registration and Authentication

- The system must allow users to register and log in securely using Clerk Authentication or Email/Google sign-in.
- Users should have personalized profiles to store interview history, feedback, and scores.
- Only authenticated users can access mock interview sessions and dashboards.

1.2 Interview Setup Module

- Users should be able to select interview type or domain (e.g., Technical, HR, Behavioral, and Aptitude).
- The system should generate a dynamic set of questions based on the chosen domain using Google Gemini API and OLLAMA.
- The user can choose between text-based and voice-based interview modes.

1.3 AI Question Generation

- The system should use Natural Language Processing (NLP) and Large Language Model (LLM) capabilities to generate relevant, context-aware interview questions.
- Questions should adapt dynamically to user responses, creating a realistic interview experience.

1.4Speech-to-Text (STT) and Text-to-Speech (TTS) Integration

- The system should convert the interviewer's questions into audio using TTS.
- The user's spoken responses should be transcribed into text using Speech-to-Text APIs.
- The system should support multiple accents and moderate background noise.

1.5Response Evaluation Module

- The system should analyze user responses using NLP and sentiment analysis to assess:
- Relevance of the answer
- Confidence and tone (via voice parameters)
- Grammar and coherence
- The evaluation should generate both a quantitative score and a qualitative explanation.

1.6 Feedback and Scoring

- After each interview, the system should provide a comprehensive feedback report showing:
- Accuracy and completeness of answers
- Communication and fluency score
- Confidence level (based on vocal cues)
- Sentiment polarity (positive, neutral, negative tone)
- Feedback should be displayed graphically (charts or progress bars).

1.7Result Visualization and Dashboard

- The user dashboard should show performance trends across multiple interviews.
- Users can compare their progress and view historical reports stored in the SQL database.

1.8 Admin/Moderator Functionality (optional)

- Admins can manage question templates, monitor user data, and view analytics.

1.9 Data Management

- All interview data, feedback, and performance metrics should be securely stored in MySQL Workbench
- Data should be retrievable for analysis and visualization.

1.10 Responsive User Interface

- The system must offer an intuitive and accessible interface, compatible with desktops, tablets, and smartphones using ReactJS and Shadcn UI.

2.Technical Feasibility

The proposed system is technically feasible due to the availability of advanced AI frameworks, cloud infrastructure, and open-source technologies.

Frontend:
Developed using ReactJS with TypeScript for speed, modularity, and responsive design.

Backend:
Implemented using Node.js and Express.js, providing asynchronous, efficient API handling.

AI Integration:
Uses Google Gemini API for question generation and Speech-to-Text/Text-to-Speech APIs for conversation simulation. Question generation and Speech-

Database:
MySQL and json token ensure secure structured data storage and scalability.

Authentication:
(JWT) handles login and authorization seamlessly.

Hosting:
Deployment through Vercel or Render ensures low-latency access worldwide.

Experimentation Details

The experimental environment, system components, datasets, test procedures, and evaluation criteria adopted to assess the effectiveness of the proposed LLM and NLP-powered Mock Interview Web Application. The setup aims to evaluate both the technical performance of the system and its impact on user interview readiness.

Participant Selection

To assess usability and effectiveness, 50 participants were recruited, including:

- Fresh graduates
- Job seekers
- Students preparing for technical placements
- Individuals with varying interview experience

Each participant provided consent before participation.

Experiment Workflow

The experimental procedure consisted of four phases:

1: Pre-Assessment

Participants completed:

- A baseline interview readiness questionnaire
- A short **self-assessment test** evaluating their communication and confidence levels

This established their starting proficiency.

2. Mock Interview Session

Participants logged into the system and selected an interview domain:

- Technical (e.g., Web Development, Data Structures)
- HR
- Behavioral
- Mixed domain

The system then initiated:

a) LLM-driven question generation
b) Real-time TTS playback of questions
c) Users responded via speech or text
d) STT transcribed audio into analyzable text
e) NLP modules evaluated:
    a. Relevance
    b. Fluency
    c. Confidence
    d. Emotional tone
    e. Completeness
    f) Responses were logged in the database for analysis.

3: Feedback Delivery

After each session, the system presented:

- A detailed performance scorecard.
- Graphical visualizations of sentiment, relevance, and fluency
- AI-generated improvement suggestions
- Recommended practice topics

4: Post-Assessment

Participants completed:

- A post-interview test to measure improvement
- A user satisfaction survey on system accuracy, usability, and realism

Implementation

1-Module-Wise Description

The AI-powered mock interview system has been designed in a modular structure, where each module performs a distinct role in the workflow.

This modularity enhances maintainability, scalability, and ease of testing. The six key modules are described below.

2-Audio Preprocessing Module

This module is responsible for preparing user speech input for analysis. The raw audio data recorded during an interview session may contain noise, silence, or variations in pitch that must be normalized before being processed by AI components.

Core functionalities:

- Noise Reduction: Filters unwanted background noise using spectral gating.
- Silence Removal: Detects low-energy segments and trims them to retain only meaningful speech.
- Volume Normalization: Equalizes the amplitude to maintain consistency across users.
- Format Conversion: Converts audio into the .wav or .flac format at a 16 kHz sampling rate for accurate Speech-to-Text (STT) processing.

3-Feature Extraction Module

After preprocessing, this module extracts meaningful linguistic and paralinguistic features from the user's responses.

These features are critical for evaluating the quality, confidence, and emotion embedded in the answer.

Extracted features include:

Linguistic Features (Text-based):

a) Keyword density
b) Sentence fluency and grammar accuracy
c) Relevance to the question context

Acoustic Features (Voice-based):

a) Pitch and tone variation
b) Speech rate (words per minute)
c) Energy levels and pauses

1.The Speech-to-Text (STT) API transcribes the candidate's response.

2.The resulting text is processed using Natural Language Processing (NLP) models for semantic analysis.

3.Parallelly, audio properties are analyzed using Librosa to detect confidence and emotion markers.

4-Model Training Module

This module builds the machine learning and AI models that power question generation, response analysis, and feedback generation.

Core Components:

Question Generation Model:

- Uses Ollama (or LLM such as GPT-4)
- Context-aware generation ensures follow-up questions are relevant to previous responses.

Response Evaluation Model:

- Uses NLP techniques such as semantic similarity and sentiment analysis to evaluate answer quality.
- Pretrained embedding from Ollama LLM are used for contextual scoring.

Scoring Engine:

- Combines metrics such as relevance, tone confidence, vocabulary richness, and fluency.
- Produces a final weighted performance score.

Algorithmic Approach:

- Supervised learning for response classification (Good, Average, Poor).
- Heuristic-based scoring for confidence and tone parameters.

Training Dataset:

- A combination of publicly available datasets (interview question-answer corpora, conversational datasets) and manually curated mock data.

Future Work

The AI Mock Interview System was successfully developed to simulate real interview environments using speech input and AI-driven analysis. The system integrates speech-to-text processing, Ollama-based feedback generation, and an interactive web interface to evaluate a candidate's communication skills, clarity, and confidence. To enhance the system's functionality, scalability, and realism, the following future improvements are proposed:

Conclusion

The AI Mock Interview System successfully bridges the gap between traditional interview preparation and intelligent digital assessment tools. By leveraging speech analysis and AI-driven feedback, it offers a realistic, efficient, and personalized interview simulation experience. The system has shown strong performance in terms of accuracy, user satisfaction, and feedback relevance, making it a promising platform for students, job seekers, and professionals aiming to enhance their communication and confidence before actual interviews. With further refinement and integration of multimodal and adaptive technologies, this project can evolve into a comprehensive virtual interview coach, capable of delivering real-time, holistic evaluations similar to those of expert human interviewers.

to dynamically generate domain-specific and difficulty-balanced inte

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., et al. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems (NeurIPS), 30.

[2] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. OpenAI Technical Report.

[3] Zhang, Y., Qin, J., Fu, J., & Zhang, Y. (2022). Automatic Evaluation of Interview Skills Using Speech and Language Features. IEEE Transactions on Affective Computing, 13(2), 745–758.

[4] DOI: 10.1109/TAFFC.2020.3005874

[5] Jauk, S., & Mair, M. (2023). AI Interview Coaching Systems: Human-Centered Design and Evaluation. International Journal of Artificial Intelligence in Education, 33(4), 1125–1143.

[6] Sujan Hiregundagal Gopal Rao. (2023). Safety and Security Co-Design in Automotive Semiconductor Systems: Challenges and Future Directions. International Journal of Intelligent Systems and Applications in Engineering, 12(4s), 830–834. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/7999

[7] Google Cloud. (2024). Speech-to-Text API Documentation.

[8] Ollama. (2024). Ollama Model Documentation.

[9] Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[10] Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations (ICLR).

[11] Abadi, M. et al. (2016). Tensor Flow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Google Research.

[12] Node.js Foundation. (2023). Node.js Documentation.

[13] Express.js Framework. (2023). Fast, unopinionated, minimalist web framework for Node.js.

[14] React.js Team. (2024). React: A JavaScript Library for Building User Interfaces. Meta Open Source.