

Design and Evaluation of an AI-Powered Interview Preparation Platform with Adaptive Question Generation and Voice-Based Interaction

Anil Kumar

R. R. Institute of Modern Technology
Lucknow, Uttar Pradesh, India

Shristi Tripathi

R. R. Institute of Modern Technology
Lucknow, Uttar Pradesh, India

Abstract—Preparing for an interview often involves searching across several websites, selecting questions that match a target role, and deciding how an answer could be improved. InterviewAI brings these activities into one web application. A user can select an interview role, interview type, difficulty level, and question count; provide resume text as context where supported; practise with generated questions; review answer feedback; and revisit earlier practice activity. The application combines a React frontend, a FastAPI backend, MongoDB storage, and a generative AI service with a predefined fallback question bank. This paper describes the system architecture, major modules, request workflow, and an evaluation plan for checking the application. It also discusses limitations related to resume extraction, variable AI output, and the need to validate feedback quality. Verified benchmark measurements and user-study results were not available when this draft was prepared; therefore, no numerical claim is made about accuracy or improvement in interview performance. The proposed tests identify practical checks and metrics that can be collected in future work.

Index Terms—Interview preparation, generative AI, question generation, resume analysis, mock interview, FastAPI, MongoDB.

I. INTRODUCTION

Interviews are an important part of recruitment, yet preparing for them can be difficult to organise. A candidate may practise from several sources without knowing whether the questions match the intended role or whether an answer covers the important points. InterviewAI was developed to make this practice process more organised: the user chooses an interview configuration, works through generated questions, receives feedback, and can return to previous practice records.

The application connects a React-based interface to backend services built with FastAPI. These services handle account access, resume text extraction, question generation, and storage of interview-related information. A predefined question bank is also available as a fallback when AI generation does not produce usable output.

This paper records the system’s design and current implementation scope. It does not claim that InterviewAI improves interview performance, because such a conclusion would re-

quire measured results from functional tests and users. The focus here is on the architecture, implementation decisions, and a practical evaluation plan.

II. PROBLEM STATEMENT AND OBJECTIVES

A candidate’s preparation can become fragmented when role-specific questions, resume review, practice answers, and feedback are handled in different places. AI generation offers a way to tailor questions, but it can also produce irrelevant content or fail when an external service is unavailable. The design problem is therefore not only to generate questions, but also to validate the output and keep the practice workflow usable when generation fails.

InterviewAI was built with six practical goals: authenticated access; text extraction from supported PDF resumes; configurable question generation; structured answer feedback and follow-up support; storage of practice history; and a fallback question bank for generation failures.

III. RELATED WORK

Question-generation research suggests that a generated question should be judged on more than whether its wording resembles a reference. QGEval, proposed by Fu *et al.*, considers dimensions including fluency, clarity, conciseness, relevance, consistency, and answerability [1]. These dimensions are useful for interview practice: a question can be grammatically correct but still be too vague, off-topic, or difficult to answer as written.

BERTScore uses contextual representations to compare generated text with reference text and can serve as one semantic similarity measure [2]. It cannot, by itself, establish that a question is useful for a particular job role or that the feedback given to an answer is correct. Human review and criteria designed for the intended task are still needed.

Work on open-domain dialogue also highlights the challenge of maintaining coherent multi-turn conversation [3]. In interview practice, a follow-up should connect to the candidate’s previous response rather than simply introduce an unrelated question. InterviewAI includes a follow-up workflow, but its relevance and consistency need to be assessed through testing.

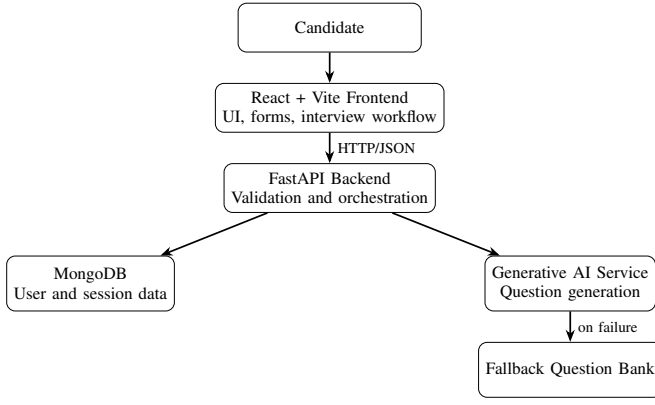


Fig. 1. High-level architecture of InterviewAI.

IV. SYSTEM REQUIREMENTS

A. Functional Requirements

The system is intended to support registration and login, authenticated user access, resume upload and text extraction, configurable question generation, answer evaluation and feedback, follow-up question generation, and interview-history retrieval. Available configuration includes multiple software-development and data-related roles, three interview types (Technical, HR, and Mixed), three difficulty levels (Beginner, Intermediate, and Advanced), and question counts of 5, 10, or 15.

B. Non-functional Requirements

The design prioritises maintainability, structured API validation, persistence of user and session information, graceful handling of AI-service failure, and protection of authentication credentials. Password hashing and token-based authentication are used in the backend design. Resume-derived text should be treated as user-provided sensitive data and should not be exposed in logs or public examples.

V. SYSTEM ARCHITECTURE

InterviewAI uses a client-server design. The React frontend presents the configuration forms, questions, feedback, and history views. FastAPI receives the requests and coordinates authentication, resume processing, question generation, and database operations. MongoDB stores the application's user and interview-related records. Question generation is handled through a backend AI service; when its response is unavailable or unusable, the application can draw questions from a predefined fallback bank.

VI. TECHNOLOGY STACK

VII. METHODOLOGY AND IMPLEMENTATION

A. Authentication and Data Validation

The backend provides registration, login, and authenticated profile operations. Passwords are hashed rather than stored as plain text, and token-based authentication is used to protect authenticated routes. Pydantic schemas validate incoming data and help keep API request and response structures explicit.

TABLE I
TECHNOLOGY STACK AND RESPONSIBILITIES

Technology	Responsibility
React, Vite, JavaScript	Client-side interface and development server
Tailwind CSS	Interface styling
Python, FastAPI	Backend APIs and request validation
MongoDB	Persistent user and interview-related records
JWT, bcrypt	Token-based authentication and password hashing
Google Gemini API	AI-assisted question generation and related text tasks
pypdf	Text extraction from supported PDF resumes
Pydantic	Request and response schema validation

B. Resume Processing

The resume workflow accepts supported PDF documents and extracts text for downstream analysis. The implementation rejects encrypted PDFs and documents that yield too little extractable text. The amount of text passed to the AI service is bounded to reduce oversized requests. PDF text extraction can fail for scanned documents or unusual layouts; optical character recognition would be a separate enhancement if required.

C. Question Generation

When a user requests questions, the backend combines the selected role, interview type, difficulty, requested count, and resume context if it is available. The AI service is expected to return structured question data, which the backend checks or normalises before sending it to the frontend. If the service fails or the response cannot be used, the fallback question bank provides an alternative source of practice questions. The fallback improves continuity, although its questions may be less tailored than a successful AI response.

D. Answer Feedback and Follow-up

After a candidate submits an answer, the platform can return structured feedback and, where configured, suggest a follow-up question based on the conversation so far. This feedback is intended to help the candidate practise; it has not been validated as a measure of job readiness and should not be used to make hiring decisions. Its accuracy, usefulness, and consistency need to be reviewed before stronger claims can be made.

E. Request Validation and Error Handling

The backend uses request schemas to check the structure and allowed values of incoming data before processing it. For question generation, the selected interview type, difficulty, and requested number of questions should be validated so that unsupported combinations can be rejected clearly. Errors from file extraction or the AI service should return a useful response without exposing credentials or internal configuration. The fallback question bank is intended to handle generation failure, but its output should still be checked for relevance and format.

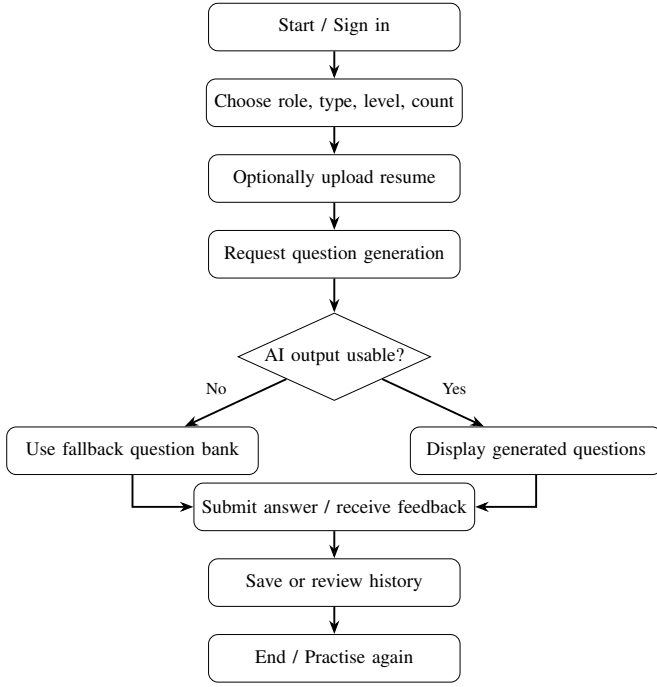


Fig. 2. Interview preparation workflow with an AI fallback branch.

F. Persistence and History

MongoDB is used to persist user and interview-related records. History endpoints allow the client to retrieve prior question-generation or interview information, enabling users to revisit practice activity. The data model and access rules should ensure that each authenticated user can access only their own records.

G. Voice-Related Workflow

The backend includes voice-session and transcript-related handling. The exact user-facing microphone capture and speech-to-text pipeline should be documented only to the extent that it is implemented and verified in the deployed application. This paper does not assume a particular speech-recognition engine or claim measured speech-recognition accuracy.

VIII. INTERVIEW WORKFLOW

The flowchart in Fig. 2 summarises the intended practice path, including the fallback branch for failed generation. It is a logical workflow representation; implementation testing is required to confirm each path in the running application.

IX. MODULE INTERACTION

The main modules exchange information through the backend rather than accessing each other directly from the browser. The frontend sends validated requests to FastAPI. Authentication checks protect user-specific operations, while the resume and question-generation services prepare inputs for the practice workflow. Results and session information are stored in MongoDB when the relevant operation requires

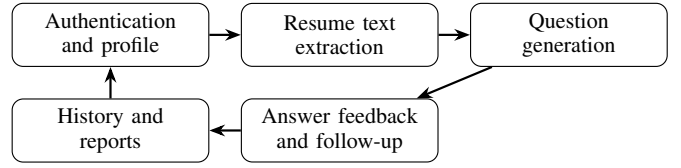


Fig. 3. Logical interaction among the main InterviewAI modules.

TABLE II
PROPOSED FUNCTIONAL TEST CASES

ID	Test	Expected outcome
T1	Register with valid details	Account created; password hashed
T2	Login with valid credentials	Authentication token returned
T3	Login with invalid credentials	Request safely rejected
T4	Upload text-based PDF	Text extracted for processing
T5	Upload encrypted/unreadable PDF	Validation error returned
T6	Generate 5, 10, and 15 questions	Requested count respected where generation succeeds
T7	Simulate AI-service failure	Fallback returns usable questions
T8	Request another user's history	Access denied
T9	Submit answer for feedback	Structured feedback returned
T10	Retrieve previous session	Correct authorised history returned

persistence. This separation makes each responsibility easier to test independently.

X. QUESTION QUALITY AND FEEDBACK CRITERIA

A meaningful evaluation should inspect question quality across several dimensions. QGEval motivates evaluating fluency, clarity, conciseness, relevance, consistency, and answerability [1]. For InterviewAI, the following application-specific checks are proposed:

- **Role alignment:** the question matches the selected role;
- **Difficulty alignment:** the question is appropriate for the requested level;
- **Resume relevance:** resume-based questions are supported by extracted content;
- **Feedback usefulness:** feedback is specific, actionable, and consistent with the answer;
- **Fallback validity:** fallback questions are relevant and well-formed;
- **Session integrity:** history and reports are associated with the correct user.

BERTScore may be included as one text-comparison metric when suitable reference questions are available, but it should not be treated as a complete measure of question quality [2].

XI. EVALUATION PLAN

The functional checks listed in Table II are a test plan, not a report of completed experiments. No verified benchmark results or participant-study data were available while preparing this draft. The tests should be run on the working application, and their outcomes recorded, before the paper is presented as an empirical evaluation.

A. Metrics to Record

For each test, record pass/fail status, environment, date, and relevant logs without storing secrets or unnecessary personal data. For generation, measure response latency over repeated trials and report sample size, median, and a percentile such as the 95th percentile. For a human quality review, use at least two independent reviewers where feasible and define a rating scale before collecting scores. Report reviewer agreement and explain the sampling procedure. Do not state accuracy, reliability, usability improvement, or candidate-outcome gains unless supported by measurements.

XII. SECURITY, PRIVACY, AND RESPONSIBLE USE

Resume documents and interview answers can contain personal information. The platform should minimise data collection, restrict access to authenticated users, avoid logging sensitive content unnecessarily, and document retention and deletion practices. API keys must remain on the server and must not be committed to source control or embedded in frontend code. Authentication, authorisation, upload validation, rate limiting, and error handling should be tested before deployment.

AI-generated feedback can be incorrect or overly confident. It should be presented as practice assistance rather than professional recruitment advice. The platform should not make employment decisions, rank applicants, or infer protected traits from voice or text. Any future voice-based scoring should be evaluated for accessibility, bias, consent, and validity before use beyond practice feedback.

XIII. LIMITATIONS

This draft describes the design and intended workflows, but it does not include measured performance results or a controlled study with users. Questions from the fallback bank may be less personalised than questions generated from a resume. PDF extraction can miss information in scanned documents or complex layouts. AI output can also be inconsistent or incorrect. In addition, backend support for voice sessions and transcripts does not by itself confirm that a complete microphone-to-transcription workflow has been tested. These points should be checked during implementation testing and user evaluation.

XIV. FUTURE WORK

Future work includes executing the proposed functional tests, conducting a structured review of generated questions and feedback, measuring response latency and fallback success, and running a usability study with informed participants. Other possible improvements include OCR for scanned resumes, stronger privacy controls, configurable data deletion, more robust monitoring, and explicit evaluation of question diversity and answerability. Voice interaction can be expanded only after its end-to-end capture and transcription components are implemented and verified.

XV. CONCLUSION

This paper presented InterviewAI as a web application for structured interview practice. Its main components include a React frontend, FastAPI services, MongoDB storage, resume text extraction, AI-assisted question generation, answer feedback, and a fallback question bank. The architecture is intended to let candidates configure practice sessions and continue practising when AI generation is unavailable. A set of functional tests and question-quality criteria has also been outlined. The next step is to run those tests and collect user feedback; until then, the platform’s effectiveness should not be described using unverified numerical claims.

ACKNOWLEDGMENT

The authors may acknowledge their project guide and institution here after confirming the preferred wording and institutional requirements.

REFERENCES

- [1] W. Fu, B. Wei, J. Hu, Z. Cai, and J. Liu, “QGEval: Benchmarking Multi-dimensional Evaluation for Question Generation,” in *Proc. 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024, pp. 11783–11803, doi: 10.18653/v1/2024.emnlp-main.658.
- [2] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating Text Generation with BERT,” in *International Conference on Learning Representations (ICLR)*, 2020.
- [3] D. Adiwardana *et al.*, “Towards a Human-like Open-Domain Chatbot,” arXiv:2001.09977, 2020.