REVA UNIVERSITY
Bengaluru, India

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A Project Report
on
AI Interviewer App

Submitted in fulfillment of the requirements for the successful completion of the course

AI Application Development (B22EFS626)

Submitted by

Deerajkumar S M(R22EF053)
Pruthvi A Gola(R22EN060)
Riya K Janali(R22EN065)

Under the guidance of

Dr. Nimrita Koul

2025

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064
www.reva.edu.in

REVA UNIVERSITY
Bengaluru, India

Certificate

This is to certify that the project titled " AI Interviewer App " is submitted by Deerajkumar S M(R22EF053), Pruthvi A Gola(R22EN060), Riya K Janali(R22EN065) in fulfilment of the requirements for the successful completion of the course **AI Application Development**.

Signature of Course Coordinator

Signature of the Director

Index

1. Abstract

This project introduces an intelligent web-based platform called the "AI Technical Interviewer," designed to simulate a comprehensive and realistic technical interview experience. The system leverages artificial intelligence to dynamically generate both theoretical and practical coding questions from various technical domains. It allows users to respond either through speech or typed input, making the interaction flexible and user-friendly. Once an answer is provided, the system uses advanced AI models to transcribe, analyze, and evaluate the response, offering immediate, detailed feedback and scoring. Visual aids such as charts and certificates enhance user motivation and progress tracking. The solution aims to provide job seekers and students with a self-paced, interactive environment to improve their technical interview skills.

2. Introduction

In the competitive landscape of technology-driven careers, acing technical interviews has become more critical than ever. Candidates are often judged not just on their knowledge but also on how effectively they can communicate their understanding and solve problems under pressure. Traditional preparation methods, including mock interviews or online problem-solving platforms, often fail to replicate the real-time dynamics and assessment rigor of actual interviews. Recognizing this gap, our project aims to fill the void by creating a virtual AI-powered interviewer that emulates a real interview scenario.

The AI Technical Interviewer provides domain-specific questions, evaluates both spoken and written responses, and gives constructive feedback in real time, thus offering users a complete preparation tool. This approach not only aids knowledge retention but also boosts confidence and performance during actual interviews.

The system introduces a new paradigm in interview preparation by integrating various advanced technologies such as machine learning, voice recognition, and natural language understanding. Unlike conventional platforms that offer question banks or peer reviews, this application simulates a one-on-one interaction with a virtual interviewer. It accommodates both theoretical and coding questions tailored to the user's selected technical domain and supports both verbal and textual modes of response.

Additionally, the platform promotes an engaging learning experience by offering personalized feedback, progress tracking through visual analytics, and performance-based certification. These features make it not only a practice tool but also a means of self-assessment and improvement. By replicating the pressure, timing, and feedback mechanisms of real interviews, the AI Technical Interviewer empowers candidates to refine their communication skills, test their problem-solving abilities, and walk into their actual interviews with greater preparedness and confidence

3. Literature Review/Study of Existing Similar Projects

A wide range of platforms currently support learners in preparing for coding interviews and improving their technical skills. These include well-established services such as **LeetCode**, **HackerRank**, and **InterviewBit**, which offer extensive libraries of coding challenges covering data structures, algorithms, and problem-solving techniques. These platforms are widely used for practice and preparation due to their structured approach, company-specific question sets, and real-time code execution features. However, they typically provide a **text-only interface**, limiting the user to written input and lacking the capability to simulate a verbal, interview-like experience.

In the realm of educational technology, **Coursera**, **edX**, and **Udacity** provide comprehensive **online courses on interview preparation**, programming fundamentals, and soft skills. Courses such as *"Technical Interviewing" by University of San Diego on Coursera* or *"Cracking the Coding Interview" on Udemy* combine lecture videos, quizzes, and assignments to help students learn at their own pace. However, these are **not interactive in real-time** and rely mostly on passive learning and asynchronous evaluations.

On the academic front, research has explored the **use of NLP for evaluating written answers**, especially in educational assessments and e-learning. Several studies have proposed automated scoring of essays and short answers using machine learning and transformer-based models. Some tools have attempted to apply these techniques in technical domains, but their scope remains limited to **textual inputs and post-hoc analysis**.

The integration of **real-time speech-to-text transcription**, **AI-driven question generation**, and **automated scoring/feedback systems** is still a relatively unexplored frontier. Most platforms tackle these functionalities in isolation, with no end-to-end system for conducting, transcribing, evaluating, and giving feedback on interviews — particularly in both **voice and text modalities**.

This project aims to address those gaps by combining multiple cutting-edge technologies into a **single, unified platform**. By leveraging **FasterWhisper** for speech recognition, **LLaMA-3** via Groq API for intelligent question generation and evaluation, and **Streamlit** for a user-friendly interface, the system simulates a real interview environment. The addition of **real-time feedback, scoring, chart visualization, and certificate generation** makes it not just a preparation tool, but a comprehensive **AI-powered interview simulation engine**.

4. Problem Statement

Despite the abundance of platforms for programming practice, there remains a significant gap in tools that simulate the complete experience of a technical interview, particularly with voice input, live feedback, and real-time evaluation. Most platforms do not support verbal responses or personalized feedback. Furthermore, they lack integrated systems that evaluate theoretical and coding knowledge with equal emphasis. Hence, the problem addressed in this project is the development of an automated AI-based technical interviewer that Generates domain-relevant theoretical and coding questions, Accepts responses via voice or text, Uses AI models to transcribe, analyze, and score answers, Uses AI models to transcribe, analyze, and score answers

4. Technology Stack Used

- **Frontend/UI**: Streamlit (Python-based web app framework)
- **Speech Recognition**: FasterWhisper (low-latency, accurate voice-to-text engine)
- **Text-to-Speech**: gTTS (Google Text-to-Speech API) integrated with pygame for audio playback
- **AI Question Generation**: LLaMA-3 model via Groq API for creating domain-specific interview questions
- **Answer Evaluation**: Groq API-based LLaMA model to provide scores and detailed feedback
- **Code Execution**: Python's subprocess module for running submitted code in Python, Java, C++, etc.
- **Data Visualization**: Matplotlib for creating score charts and progress visuals
- **Certificate Generation**: PIL (Python Imaging Library) to create downloadable certificates
- **File Handling**: UUID, datetime, and OS libraries to log user data and save transcripts

5. System Architecture

The system architecture comprises several interconnected modules that together create an interactive, intelligent interview environment:

- **User Interface Module**: Developed using Streamlit, it provides input forms, buttons, and display sections for questions, answers, scores, and results.
- **Voice Engine Module**: Records audio responses using sounddevice and transcribes them using FasterWhisper.
- **Question Generator**: Leverages LLaMA-3 via Groq to create meaningful, context-aware questions in theoretical and coding formats.
- **Evaluator Module**: AI models assess the relevance and quality of user responses, providing scores and improvement suggestions.
- **Code Execution Engine**: Executes submitted code using subprocess in a secure temporary environment and returns outputs/errors.
- **Results Processor**: Aggregates scores, visualizes performance, assigns badges, and generates certificates.
- **Data Logging Module**: Saves session logs, transcripts, and feedback to enable progress tracking and future reference.

6. Methodology / Workflow

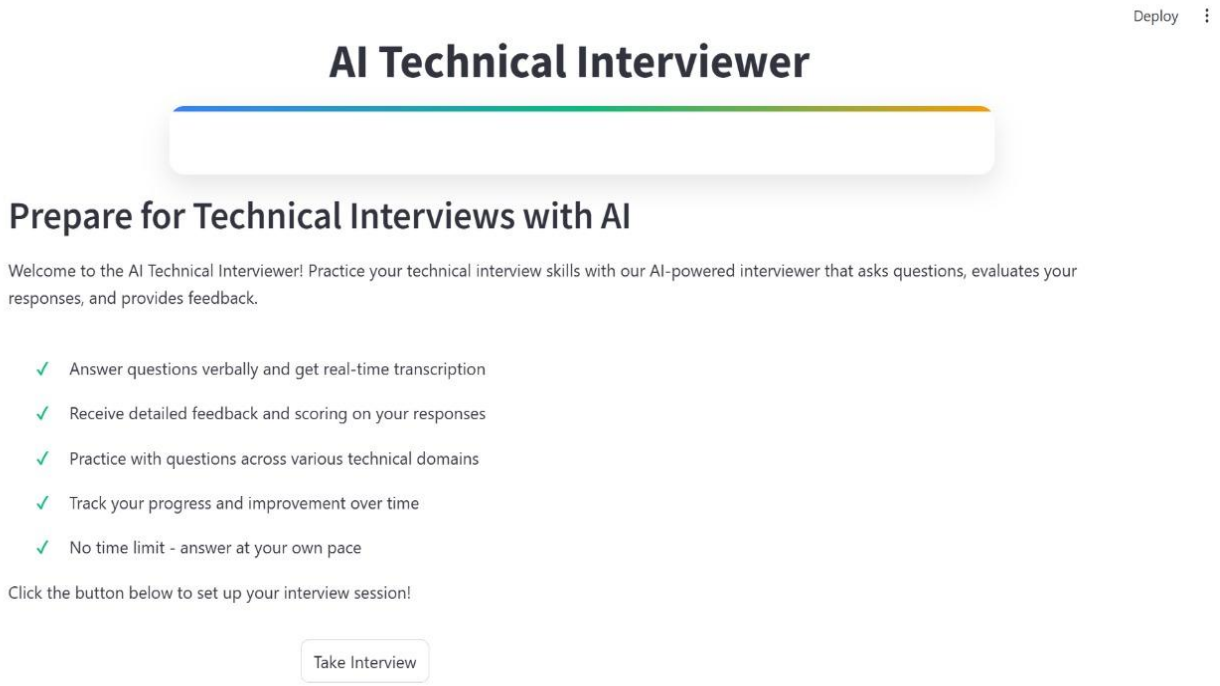
The workflow followed by the AI Technical Interviewer application includes:

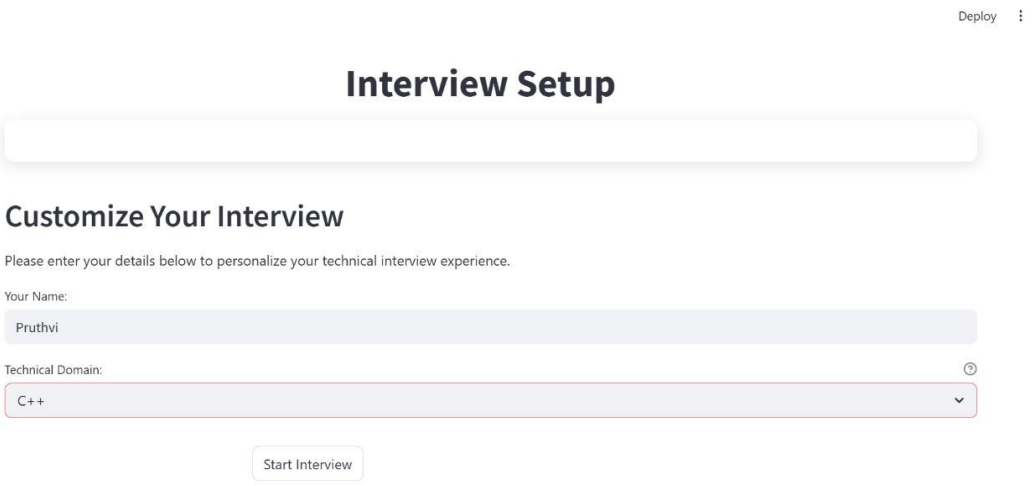
- **Interview Setup**: Users enter their name and choose a domain (e.g., Python, Java, Data Science).
- **Question Generation**: AI generates a relevant theoretical or coding question based on the chosen domain.
- **Answer Submission**:
    - Voice-based: User records an answer which is transcribed by FasterWhisper.
    - Text-based: User types the answer directly into the provided interface.
- **Answer Evaluation**: The AI checks for relevance, scores the answer, and provides constructive feedback.
- **Coding Question Execution**: For code-based questions, user submits code which is run and evaluated.
- **Feedback Display**: Scores, strengths, weaknesses, and improvement areas are shown.
- **Score Visualization**: A donut chart shows average performance, and a badge is awarded.
- **Certificate Generation**: Based on performance, a downloadable certificate is issued.
- **Navigation**: User can continue to the next question or end the interview.

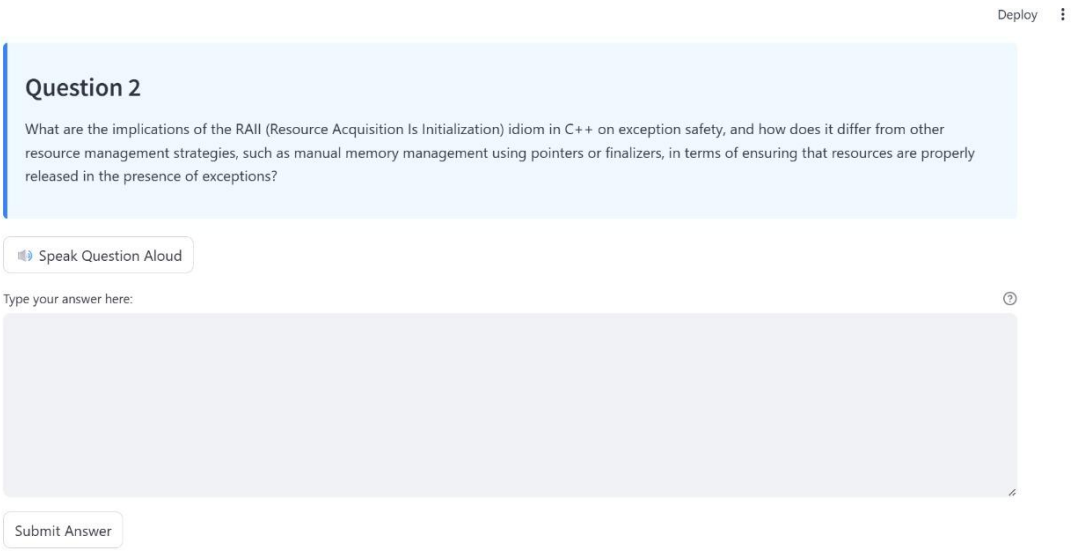
7. Implementation and Screenshots

The implementation is fully done in Python using the Streamlit framework. The app is structured to support modularity and real-time interaction. Key implementation highlights:

- Use of threading to allow asynchronous audio playback
- Groq API integration for high-quality LLaMA-based response evaluation
- Matplotlib donut charts and animated confetti for results display
- Voice recording using sounddevice and scipy.io.wavfile
- Secure and temporary code execution using Python subprocess







Feedback:

Score: 8

Strengths:

- The answer clearly defines RAII and explains its role in exception safety, providing a solid foundation for understanding the concept.

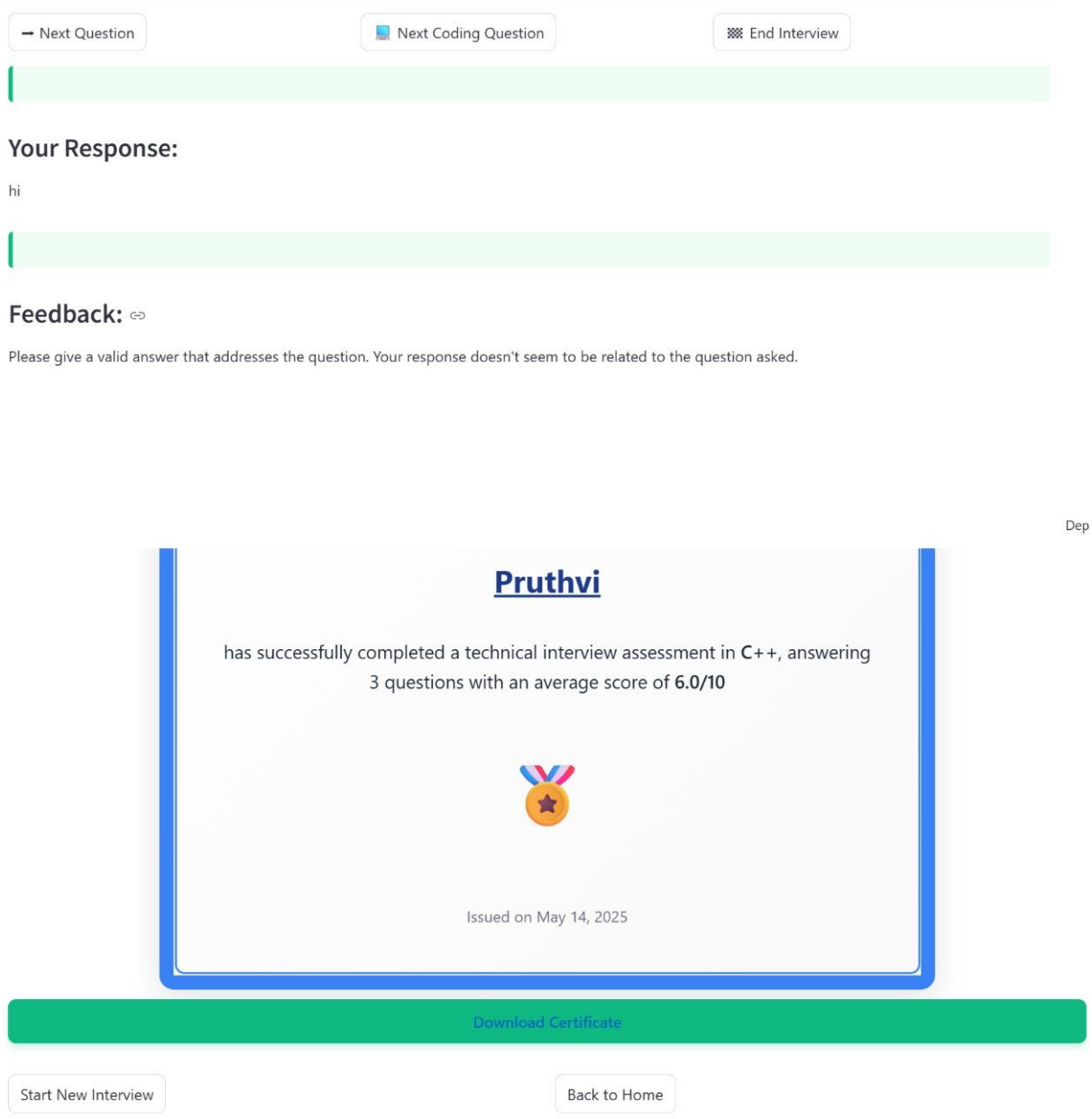
- The answer breaks down the exception safety guarantees (basic, strong, and no-throw) and connects them to the use of RAII, demonstrating a good understanding of the topic.

Areas for Improvement:

- The answer could benefit from more specific examples or code snippets to illustrate how RAII is implemented in practice, making the concept more concrete for readers.

- The comparison to other resource management strategies (such as manual memory management using pointers or finalizers) is mentioned but not fully explored, leaving an opportunity to provide a more comprehensive analysis of the trade-offs between these approaches.

Overall Feedback: The answer provides a clear and concise overview of RAII and its implications for exception safety in C++. To further enhance the response, consider adding concrete examples or code snippets to demonstrate the application of RAII, and expand on the comparison to other resource management techniques to give readers a more nuanced understanding of the trade-offs involved. With these additions, the answer could be even more effective in educating readers about the importance and implementation of RAII in C++.


Next Question
Next Coding Question
End Interview
Your Response:
hi
Feedback:
Please give a valid answer that addresses the question. Your response doesn't seem to be related to the question asked.
Dep
Pruthvi
has successfully completed a technical interview assessment in C++, answering 3 questions with an average score of 6.0/10
Issued on May 14, 2025
Download Certificate
Start New Interview
Back to Home

8. Results

The system performs efficiently across all functions:

- Accurate transcription with minimal delay
- Fast and meaningful question generation
- Rich, personalized feedback with scoring for each answer
- Live progress visualization and session tracking
- Interactive and aesthetically pleasing user interface

Users are able to complete mock interviews independently, receive objective assessments, and identify areas for improvement. Certificates and badges increase motivation.

9. Challenges

- Speech Recognition Accuracy: Adapting the system for different accents and noise levels required tuning.
- Groq API Limitations: Managing rate limits and timeouts for large volumes of text.
- Code Execution Security: Running arbitrary code in a secure yet efficient way was complex.
- Parsing Feedback: Extracting structured data (scores, feedback sections) from LLM outputs required regex and fallback strategies.
- UI Responsiveness: Managing Streamlit state across multiple user actions while keeping UI intuitive.

10. Conclusion and Future Work

The AI Technical Interviewer successfully simulates a real-world technical interview environment, helping users build confidence and technical proficiency. It integrates several complex technologies into a unified platform that is easy to use and highly effective.

Future Enhancements:

- Support for video answers and emotion analysis
- More advanced analytics with historical performance tracking
- Multiplayer/mock panel interview simulations
- Resume analysis and job-fit recommendations
- Offline mode and mobile compatibility

11. References

- OpenAI LLaMA : https://ollama.com/
- Groq API: https://console.groq.com/keys
- Gtts: https://pypi.org/project/gTTS/
- Matplotlib: https://matplotlib.org/