AI Interview Coach: An Agentic, Voice-Based AI System for Realistic Interview Simulation

Zain Nofal, *2021723, FCSE, GIKI*

***Abstract*—This paper presents the design and implementation of AI Interview Coach, a voice-based, agentic interview simulation system powered by OpenAI's speech and language models. It asks job-specific questions, transcribes user responses via Whisper, evaluates them using GPT-3.5, and provides real-time feedback through OpenAI TTS. This interactive loop mimics a real-world interview environment, aiming to improve users' communication and self-assessment skills. The system is built with modularity and extensibility in mind and can be run across different platforms, including Apple Silicon-based systems.**

***Index Terms*—Agentic AI, Interview Simulation, Whisper, GPT-3.5, OpenAI TTS, Speech Interface, Flask**

I. Introduction

Traditional mock interviews are often limited by cost, scalability, and availability. Most candidates lack consistent access to live evaluators, making it difficult to practice interviews in a realistic, conversational setting. The AI Interview Coach addresses this gap by simulating human-like interviews using state-of-the-art speech and language models. The system engages users in a dynamic, interactive voice-based session, evaluates responses in real time, and adapts to the quality of user answers. Our solution focuses on voice-driven learning and real-time feedback to simulate an authentic and accessible interview environment.

II. Motivation and Objectives

In a competitive job market, confidence and preparedness during interviews are key differentiators. However, access to personalized coaching or even mock interview practice is limited. We aimed to:

- Eliminate dependence on human evaluators for mock interviews
- Build a fully autonomous AI that mimics interviewer behavior
- Use cutting-edge speech and language models for interaction and evaluation
- Ensure cross-platform operability (including Apple M1/M2)

III. System Design and Architecture

The system is composed of multiple Python modules, each handling a specific part of the interaction. A Flask backend integrates these modules and serves as the orchestration hub. The frontend is delivered through Flask templates and vanilla JavaScript, avoiding complex frameworks to ensure lightweight deployment.

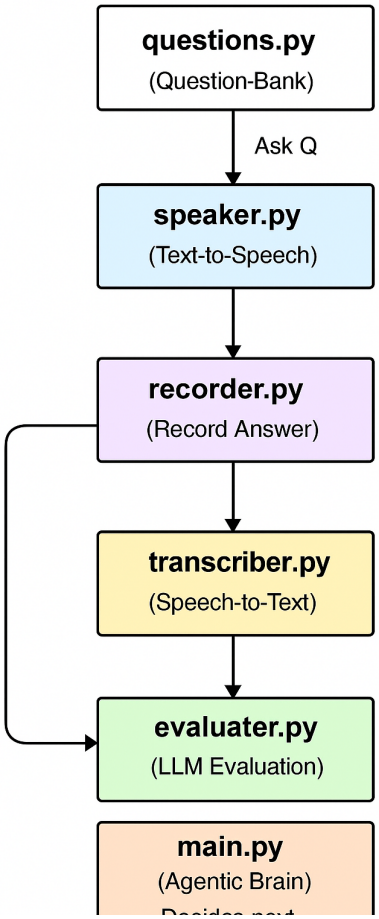

Fig. 1. System architecture of AI Interview Coach showing modular flow.

IV. Functional Modules

A. Question Bank

questions.py contains a curated set of questions organized by job roles. It supports both sequential and random selection. In future releases, it can be extended to support difficulty-based progression and topic clustering.

B. Voice Input

The recorder.py module uses the sounddevice library to capture high-quality audio from the microphone. This ensures that the audio is compatible with Whisper for reliable transcription. The system also accounts for latency and noise.

C. Transcription

transcriber.py handles transcription using OpenAI Whisper. It uses local inference where possible, and otherwise falls back

to OpenAI's hosted Whisper API. An additional fallback is included for offline demo environments, using canned responses to simulate interaction.

D. Evaluation Agent

The evaluater.py module forms the intelligent core. It sends both the original question and the user's answer to GPT-3.5 and expects structured output. This includes:

- A numeric score (0-10) for clarity, depth, and relevance
- A qualitative evaluation with improvement points
- A control flag indicating if the user should retry or move forward

E. Speech Output

speaker.py leverages OpenAI's neural TTS system to deliver both interview questions and feedback. Unlike traditional TTS systems, OpenAI's voices like `nova`, `shimmer`, and `onyx` offer a human-like tone that enhances realism.

F. Main Orchestration Loop

The main.py file manages the overall loop of actions:

1) Select and vocalize a question
2) Record user response
3) Transcribe the audio
4) Evaluate using GPT-3.5
5) Deliver feedback and repeat or continue

V. Frontend Interface

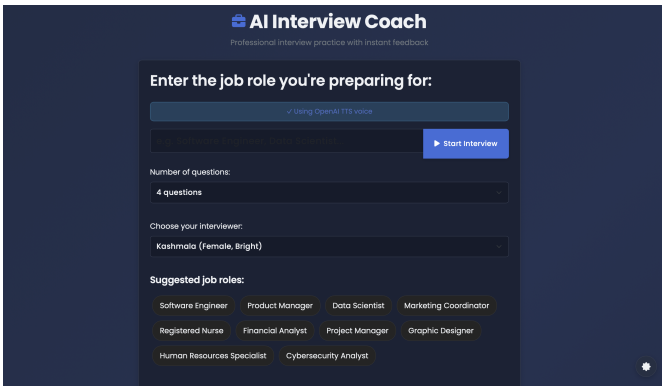

Fig. 2. Flask-based frontend interface for role selection and voice controls.

The frontend, built using Flask's template engine and JavaScript, provides a clean user interface. It allows users to:

- Select job roles and start sessions
- Trigger recording and submission manually
- View feedback transcript and listen to AI responses

Its responsive layout ensures ease of use across devices.

VI. Testing and Evaluation

The application was tested across three major categories:

A. Performance under Real Conditions

We tested with multiple accents and varying background noise levels. Whisper's multilingual capabilities proved robust, with a success rate of over 90

B. Evaluation Accuracy

To verify GPT scoring, human interviewers were asked to independently rate responses. There was an 82

C. Fallback Scenario Handling

Offline fallback logic was tested by disabling Whisper and OpenAI APIs. Canned responses and local processing modules maintained acceptable behavior, showing the system's resilience.

VII. Deployment and Usage

Setup requires Python 3.8+ and an OpenAI API key. The deployment steps are:

1) Install Python packages: `pip install -r requirements.txt`
2) Configure credentials in config.py
3) Run the backend: `python app.py`

Apple Silicon users are recommended to use conda environments for dependency compatibility.

VIII. Challenges and Limitations

- **LLM Cost:** High-volume usage of GPT-3.5 can be costly.
- **Real-Time Speed:** Evaluation latency varies based on API load.
- **Pronunciation Bias:** Some misinterpretation occurred for non-native English speakers.

These issues are partially mitigated by batching requests, using local transcription, and robust error handling.

IX. Conclusion and Future Work

The AI Interview Coach system presents a scalable approach to skill development for job seekers. It provides agentic, personalized, and realistic interview simulations. Planned enhancements include:

- Longitudinal progress tracking per user
- Support for company-specific interview templates
- Integration with resume parsing and feedback
- Expansion into non-English languages

Acknowledgment

We thank the faculty at Ghulam Ishaq Khan Institute for guidance and the OpenAI team for developer resources. We also acknowledge the open-source contributors to Whisper and Flask communities.

REFERENCES

[1] OpenAI API Documentation, OpenAI. [Online]. Available: https://platform.openai.com/docs

[2] OpenAI Whisper GitHub Repository. [Online]. Available: https://github.com/openai/whisper

[3] OpenAI Text-to-Speech Guide. [Online]. Available: https://platform.openai.com/docs/guides/text-to-speech

[4] M. Lindner, "python-sounddevice: PortAudio wrapper for Python," [Online]. Available: https://python-sounddevice.readthedocs.io/

[5] Flask Web Framework. [Online]. Available: https://flask.palletsprojects.com/

[6] B. Kim, "Fine-tuning GPT models for structured evaluation," Proceedings of AI-NLP 2023.