A person looking at a computer screen

Description automatically generated

***Interview Bot: AI chatbot for mock interviews, response evaluation, and feedback metrics.***

**Comp 385 -AI Capstone Project Report (SEC. 002)**

Date: April 14, 2025

Final Version

Submitted by:

**Lakshay (301279369)**

**Nolan Garcia Teini (301263505)**

**Rani Krishna Pillai (301302017)**

**Srinivasa Sarangam (301308754)**

**Harsh Subhash Sharma (301303949)**

**Executive Summary**

In today’s competitive job market, candidates often face challenges in preparing for technical interviews due to lack of tailored practice, real-time feedback, and access to expert guidance. Our capstone project — **AI-Powered Multilingual Interview Bot** — addresses this gap by offering an intelligent, interactive, and voice-enabled platform that simulates real-world interviews across multiple languages.

The system leverages a combination of **Natural Language Processing (NLP)**, **Speech Recognition**, and **Machine Learning (ML)** to provide a seamless interview experience. Users upload their resume and a job description, based on which the system extracts key skills using **AWS Comprehend**. These skills are mapped to a curated dataset of over 1,000 technical, behavioral, and situational questions. Users can choose their preferred language (English, French, Hindi, Spanish, or German) and answer via voice or text.

The bot then utilizes **OpenAI GPT-3.5** to generate structured feedback for each answer, along with a reference response, confidence level (via voice analysis using **Librosa**), and targeted suggestions for improvement. All interview sessions are saved with a unique session ID, allowing users to revisit and reflect on their performance.

We implemented **Gradio** as the UI framework to create a clean, chatbot-style interface. Supporting AWS services include **Polly** for text-to-speech and **Whisper** for transcribing multilingual voice responses. The system runs locally and avoids expensive inference costs by balancing cloud APIs with efficient Python-based preprocessing.

This project demonstrates the effective integration of cloud computing, AI services, and real-time NLP systems in a practical and scalable solution. It highlights our team’s ability to build end-to-end ML solutions while focusing on usability, accessibility, and real-world application.

Contents

[Introduction 4](#_Toc195489155)

[Summary of Literature Review 5](#_Toc195489156)

[Summary of Exploratory Data Exploration 5](#_Toc195489157)

[Methodology 7](#_Toc195489158)

[Research & Requirement Gathering 7](#_Toc195489159)

[Exploratory Data Collection & Generation 7](#_Toc195489160)

[AI Capability Development – Iteration 1 8](#_Toc195489161)

[AI Capability Refinement – Iteration 2 8](#_Toc195489162)

[AI Capability Refinement – Iteration 3 (Final Iteration) 8](#_Toc195489163)

[System Design & Architecture 9](#_Toc195489164)

[Full Stack Development 10](#_Toc195489165)

[Testing 11](#_Toc195489166)

[Results / Data / Analysis 12](#_Toc195489167)

[Conclusions 13](#_Toc195489168)

[Recommendations 14](#_Toc195489169)

[Bibliography 15](#_Toc195489170)

[Appendix 16](#_Toc195489171)

[Stakeholder Register 16](#_Toc195489172)

[System Specification (Requirements Model) 17](#_Toc195489173)

[Use Case List: 17](#_Toc195489174)

[Use Case Diagram: 18](#_Toc195489175)

[Detailed Use Case Descriptions 20](#_Toc195489176)

[Functionals Requirements 21](#_Toc195489177)

[Non-Functional Requirements 21](#_Toc195489178)

[System Design 22](#_Toc195489179)

[List of Technologies 22](#_Toc195489180)

[AI Capability Description & Data 23](#_Toc195489181)

[Architecture Design – AI Capability 24](#_Toc195489182)

[Architecture Design – Full Stack 25](#_Toc195489183)

[Class Diagram 26](#_Toc195489184)

[Component Diagram 27](#_Toc195489185)

[Interaction Diagram 28](#_Toc195489186)

[Project Plan 30](#_Toc195489187)

[Test Plan and Test Cases 31](#_Toc195489188)

[UI/UX Design 34](#_Toc195489189)

[Deployment Strategy 36](#_Toc195489190)

[High – Level ML/Ops Description 37](#_Toc195489191)

[List of Acronyms 38](#_Toc195489192)

[Glossary of Terms 39](#_Toc195489193)

# List of Illustrations

Figure 1: Architecture Diagram

Figure 2: Frontend

Figure 3: pyTest coverage

Figure 4: Use case 1

Figure 5: Use case 2

Figure 6: Use case 3

Figure 7: Use case 4

Figure 8: Architecture Design

Figure 9: Full Stack

Figure 10: Class Diagram

Figure 11: Component Diagram

Figure 12: Interaction Diagram 1

Figure 13: Analyze Input

Figure 14: Generate Feedback

Figure 15: Feedback Plan

Figure 16: Test Coverage

Figure 17: UI/UX

Figure 18: ReadMe file for running code

Figure 19: Feedback and Translation Process

# Introduction

The rise of artificial intelligence (AI) and automation has significantly impacted how job interviews are conducted, evaluated, and practiced. While there has been growth in online platforms offering interview preparation tools, many lack personalization, real-time feedback, and the integration of modern NLP and voice technologies. As job candidates increasingly seek meaningful, customized practice experiences, it becomes essential to provide tools that simulate interviews in a natural and intelligent way.

The issue this project addresses is the **absence of an interactive, multilingual interview simulation platform** that leverages AI to assess both technical knowledge and soft skills using natural voice input. Current interview platforms focus either on static question banks or limited chatbot experiences, often in English only and without meaningful feedback. Our solution aims to provide a smarter, adaptive alternative.

This problem is significant in the context of job readiness and AI-driven skill assessment. Students, new graduates, and even experienced professionals often feel underprepared for technical and behavioral interviews. By combining real-time question generation, multilingual support, speech analysis, and GPT-based feedback, our project fills an important gap in accessible AI-powered career preparation tools.

Summary of Literature Review**:**

Various tools and models have been developed to support AI-based tutoring and language processing. Solutions such as Pramp and Interviewing.io offer mock interview setups but often lack multilingual capabilities and voice interaction. Research on conversational agents in recruitment (Gnewuch et al., 2020) shows potential for AI in interview contexts, but many commercial tools still lack integration of GPT-level reasoning, voice-to-text transcription, and personalized content based on resume parsing.

Studies also emphasize the value of **automated feedback**, especially in language learning and soft-skill training (Xie et al., 2022). However, few solutions integrate **speech confidence analysis**, **multilingual translation**, and **tailored question selection** in one package. Our project combines these components in a novel way.

Summary of Exploratory Data Exploration**:**

Since no public dataset directly met our requirements, a **synthetic dataset of over 1,000 question-answer pairs** was generated using tools such as **GitHub Copilot**, **Claude**, and **manual prompting with OpenAI GPT**. The dataset includes:

* 800+ technical questions across backend, frontend, and cloud domains
* 100 behavioral questions
* 100 situational questions

Questions were refined to match real-world interview phrasing, and many were inspired by job postings sourced from portals like **LinkedIn**, **Glassdoor**, and **Indeed**. Skill labels were manually reviewed and assigned to ensure compatibility with resume-sourced keywords.

The synthetic data was structured in JSON format and aligned with experience levels (entry, mid, senior) and skill domains (e.g., Python, Java, Cloud, Agile). The dataset forms the backbone of our system's interview generation logic.

# Methodology

The purpose of this methodology is to provide a clear, structured explanation of how the proposed AI Interview Bot was designed and developed. The approach integrates research, data acquisition, iterative AI model development, system design, and testing. All project phases closely follow the project plan detailed in the appendix.

## Research & Requirement Gathering

We began by conducting a literature review to understand the current landscape of AI-driven interview preparation tools. Through this review, we identified gaps in voice support, feedback personalization, and multilingual capabilities.

Concurrently, we outlined the project proposal and scope, identifying key stakeholders and specifying functional and non-functional requirements. These activities informed the preparation of our requirement specification document and use case definitions.

## Exploratory Data Collection & Generation

Initial development focused on implementing AI-driven components:

* Integrated OpenAI Whisper for multilingual speech-to-text transcription.
* Implemented resume and job description parsers.
* Extracted skills using SpaCy and keyword matching logic.
* Developed a system to retrieve appropriate questions using vector similarity (SBERT).
* Generated reference answers and GPT feedback.

The goal was to build the AI backend logic and validate end-to-end capability in English.

## AI Capability Development – Iteration 1

Initial development focused on implementing AI-driven components:

* Integrated OpenAI Whisper for multilingual speech-to-text transcription.
* Implemented resume and job description parsers.
* Extracted skills using SpaCy and keyword matching logic.
* Developed a system to retrieve appropriate questions using vector similarity (SBERT).
* Generated reference answers and GPT feedback.

The goal was to build the AI backend logic and validate end-to-end capability in English.

## AI Capability Refinement – Iteration 2

We applied design patterns to optimize code structure and improve modularity. Voice confidence scoring was added to enhance feedback. Polly was integrated for multilingual TTS playback.

Internal unit tests were created for all utilities (translation, parsing, feedback) to ensure robust output before full stack development.

## AI Capability Refinement – Iteration 3 (Final Iteration)

After Iteration 2, the system underwent significant refinement through modular restructuring and the implementation of reusable utility patterns. A voice confidence analyzer was integrated to evaluate user responses more effectively. We enhanced user accessibility by incorporating AWS Polly for dynamic multilingual text-to-speech playback.

To ensure system reliability, we implemented unit tests for critical modules including resume parsing, translation, Whisper-based transcription, and GPT-driven feedback generation. These tests validated functionality before proceeding to full-stack deployment, ensuring a robust and maintainable system.

## System Design & Architecture

Architecture diagrams and flowcharts were created to represent system components. A modular design was adopted using utility scripts for translation, transcription, resume parsing, question retrieval, and voice analysis.

Data flow between components was visualized and finalized before moving to the second iteration.

A diagram of a software system

AI-generated content may be incorrect.

Figure 1: Architecture Diagram

## Full Stack Development

Using Gradio, a conversational web-based interface was created:

* File upload for resumes (PDF/DOCX)
* Textbox input for job descriptions
* Chatbot for voice/text Q&A
* Voice input for answers with multilingual transcription
* GPT feedback with scoring and session tracking

A screenshot of a computer

AI-generated content may be incorrect.

Figure 2: Frontend

Sessions could be saved and retrieved using session IDs. Audio player functionality was added to replay each question using AWS Polly.

## Testing

The testing phase included:

* Unit testing individual utilities
* Integration testing across modules
* Manual UI testing for chatbot
* Edge case testing for empty files, non-English input, and multiple answer types

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3: pyTest coverage

Code coverage was evaluated using pytest-cov, and a final report was generated showing 57% coverage across utility scripts and tests.

# Results / Data / Analysis

The system's performance and behavior were evaluated across multiple test runs using a synthetically generated dataset. This dataset was designed to simulate realistic job applications by combining job descriptions and resume content sourced and adapted via tools such as GitHub Copilot, Claude, and various online job portals. The goal was to approximate real-world user interactions in the absence of publicly available structured datasets for resume-question-answering systems.

The key findings and observations from the evaluation are summarized below:

* **Significance of Skills-Based Question Retrieval:**  
  The system successfully extracted and matched technical, behavioral, and situational questions based on key skills detected in the resumes. This skill-based mapping proved essential to maintaining interview relevance. The most accurate results were observed when resumes included clearly labeled sections such as "Skills," "Projects," and "Experience." Less structured resumes resulted in less precise matches.
* **Positive vs. Negative Outcomes:**  
  The integration of AWS Comprehend, and the in-house skill extraction logic yielded strong results (≥90% accuracy in extracting intended keywords). However, OpenAI Whisper sometimes misinterpreted low-quality audio or accented input, which introduced minor errors in transcription accuracy (average accuracy dropped to ~82% in such cases).
* **Surprising Insights:**  
  One surprising finding was the system's resilience when provided only partial information in resumes. Even with limited input, the GPT-based feedback generation and Pinecone retrieval system could construct semi-relevant interview questions, demonstrating strong generalization.
* **Multilingual Performance:**  
  With AWS Translate and Polly integrated, users were able to conduct interviews in Hindi, French, Spanish, and German with above-average clarity. However, feedback from GPT remained English-only in this iteration, highlighting a current limitation that was addressed in future refinements.
* **Testing Results:**  
  We conducted unit tests for each module (translation, transcription, parsing, and feedback). The test coverage report showed 57% overall coverage, with critical paths such as skills\_extractor.py, question\_retriever.py, and comprehend.py achieving 95–100%. Lower coverage in modules like voice analyzer was due to dependencies on local audio input, which are harder to mock.

# Conclusions

The AI Interview Bot project successfully addressed the problem of delivering personalized, skill-driven interview simulations through a multilingual, voice-enabled interface. The results demonstrated that the system could extract relevant skills from uploaded resumes, generate appropriate questions, and provide intelligent feedback using modern NLP and cloud services.

The integration of AWS Comprehend enabled accurate keyword extraction from resumes, while OpenAI GPT generated contextual, structured feedback that aligned with industry-relevant standards. The Whisper model handled transcription effectively in most test scenarios, and AWS Translate and Polly allowed support for non-English interviews, significantly increasing accessibility.

The modular architecture and layered design allowed for smooth data flow between user input, processing logic, and external services. This design not only improved maintainability but also simplified unit testing and system debugging. Internal test coverage reached 57%, with mission-critical utilities achieving over 90% accuracy and reliability.

A key conclusion is that skill extraction, even from partially structured resumes, is highly feasible and can guide automated question retrieval. Another conclusion is that the clarity and usefulness of GPT-generated feedback remain high, even without multilingual output in this iteration.

In summary:

* The system meets the intended goal of simulating intelligent interviews with dynamic voice and text input.
* Modular architecture and service integration ensured robust performance and extensibility.
* Remaining limitations—such as limited multilingual feedback—can be resolved in future iterations with expanded LLM translation and voice support.

These conclusions are supported by the data collected during testing and validate the AI Interview Bot’s potential for real-world deployment in HR technology or EdTech solutions.

# Recommendations

 **Integrate multilingual GPT feedback generation**  
Future iterations should include translated feedback using AWS Translate or multilingual LLMs, ensuring end-to-end native language support for non-English users.

 **Deploy a production-ready web interface**  
While the current system uses Gradio on localhost, migrating to a full web frontend (e.g., React or Streamlit Cloud) would enable global accessibility and user account features.

 **Expand dataset for better generalization**  
The current dataset is synthetically generated. Future development should involve curated real-world resumes and job descriptions to improve system accuracy and realism.

 **Add iOS support and mobile interface**  
To reach more users, building a mobile interface for Android and iOS platforms would allow voice-based interviews to on-the-go.

 **Enhance voice confidence scoring model**  
The existing voice confidence analyzer can be improved by training on a labeled dataset with annotated speech patterns, ensuring more accurate detection of hesitation or confidence.

 **Incorporate real-time interviewer bot persona**  
Adding an animated or voice-backed interviewer persona could enhance user engagement and simulate a more realistic interview experience.

 **Allow recruiter/admin dashboards**  
Future versions could include dashboards for HR professionals to view candidate performance summaries and feedback in real time.

# Bibliography

 Amazon Web Services. (2023). *Amazon Polly developer guide*. <https://docs.aws.amazon.com/polly>

 Amazon Web Services. (2023). *Amazon Translate documentation*. <https://docs.aws.amazon.com/translate>

 Biewald, L. (2020). *Machine learning system design*. MLRepa. <https://www.mlr.pub/>

 GitHub. (2023). *GitHub Copilot: Your AI pair programmer*. <https://github.com/features/copilot>

 Google Cloud. (2022). *Text-to-speech and speech-to-text APIs*. https://cloud.google.com/docs

 Indeed. (2023). *Sample resumes and job descriptions*. <https://www.indeed.com>

 LinkedIn. (2023). *Job listings and resume samples*. <https://www.linkedin.com>

 OpenAI. (2023a). *GPT-3.5 API documentation*. <https://platform.openai.com/docs/guides/gpt>

 OpenAI. (2023b). *Whisper speech-to-text API guide*. <https://platform.openai.com/docs/guides/speech-to-text>

 Stack Overflow. (2023). *Developer discussions on resume parsing and interview systems*. <https://stackoverflow.com/>

 Williams, A. (2021, May 14). *Understanding DynamoDB design patterns*. AWS Architecture Blog. <https://aws.amazon.com/blogs/database/>

# Appendix

## Stakeholder Register

The stakeholder register identifies key individuals and organizations involved in or affected by the project, outlining their roles, responsibilities, and levels of interest. This document ensures clear communication and collaboration by categorizing stakeholders as internal or external and detailing their positions and contact information. For this project, stakeholders range from sponsors, who provide funding and executive oversight, to internal team members, such as AI developers and project leaders, who execute the project’s technical and operational tasks. The register ensures that each stakeholder’s level of interest and involvement is clearly defined, facilitating effective project management and decision-making.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stakeholder Name** | **Position (Role)** | **External/**  **Internal** | **Stakeholder Contact Details** | **Operational/Executive** | **Interest (high, medium, low)** |
| **Job Applicants** | End Users | External | [N/A] | Operational | High |
| **Recruitment Agencies** | Hiring Partner | External | [N/A] | Operational | High |
| **Software Developers** | AI Engineers | Internal | [N/A] | Operational | High |
| **HR Experts** | Interview Specialists | External | [N/A] | Operational | High |
| **Training Providers** | Career Consultants | External | [N/A] | Operational | Medium |
| **Product Owner** | Project Manager | Internal | [N/A] | Executive | High |
| **IT Security** | Security Administrator | Internal | [N/A] | Executive | Medium |
| **Investors** | Financial Supporter | External | [N/A] | Executive | High |

## System Specification (Requirements Model)

### Use Case List:

|  |  |  |
| --- | --- | --- |
| Use Case Name | Actors | Description |
| Upload Resume | Candidate | Candidate uploads a resume in PDF or DOCX format for analysis. |
| Submit Job Description | Candidate | Candidate enters or pastes a job description into the system. |
| Start Interview | Candidate | System analyzes inputs and begins interview by presenting questions. |
| Answer Question | Candidate | Candidate responds to each question via voice or text input. |
| Receive Feedback | Candidate | Candidate receives GPT-based feedback at the end of the session. |
| Retrieve Session | Candidate | Candidate can retrieve previous session using session ID. |
| Analyze Input (AI) | System (AI Module) | The system extracts skills from resume/JD and generates appropriate questions. |
| Generate Feedback (AI) | System (AI Module) | GPT model evaluates answers and returns structured feedback. |

### Use Case Diagram:

Use Case 1: Process Uploaded Resume for Skill Extraction

A diagram of a process

AI-generated content may be incorrect.

Figure 4: Use case 1

Use Case 2: Generate Personalized Interview Questions

A diagram of a process

AI-generated content may be incorrect.

Figure 5: Use case 2

Use Case 3: Conduct Voice-Based Interview Simulation

A diagram of a speech processing system

AI-generated content may be incorrect.

Figure 6: Use case 3

Use Case 4: Generate Text-Based AI Feedback for User

A diagram of a diagram

AI-generated content may be incorrect.

Figure 7: Use case 4

### Detailed Use Case Descriptions

Use Case: Generate Feedback (AI)

| **Element** | **Description** |
| --- | --- |
| Use Case Name | Generate Feedback |
| Actors | System (GPT Module, Voice Analyzer) |
| Trigger | All user answers submitted |
| Description | System translates answers, feeds them to GPT, retrieves reference answers, and evaluates user response with feedback and confidence score. |
| Pre-Conditions | All questions must be answered |
| Post-Conditions | Feedback summary is displayed to user and saved to session |
| Alternative Flows | None if GPT API fails, fallback message displayed |

Use Case: Analyze Input (AI)

| **Element** | **Description** |
| --- | --- |
| Use Case Name | Analyze Input |
| Actors | System (AI Module) |
| Trigger | Resume and Job Description are submitted |
| Description | The system extracts skills using NLP from the uploaded resume and job description. It matches these to a dataset to select appropriate questions. |
| Pre-Conditions | Resume and JD must be uploaded |
| Post-Conditions | Question list is generated and stored |
| Alternative Flows | If skill extraction fails, fallback to general questions |

### Functionals Requirements

| **ID** | **Requirement** |
| --- | --- |
| FR1 | System shall allow users to upload resumes in DOCX or PDF format. |
| FR2 | System shall allow users to input job descriptions via text box. |
| FR3 | System shall extract skills from both resume and JD using AWS Comprehend. |
| FR4 | System shall retrieve 3 technical, 1 behavioral, and 1 situational question. |
| FR5 | System shall accept both voice and text responses from the user. |
| FR6 | System shall transcribe voice answers using OpenAI Whisper. |
| FR7 | System shall evaluate answers using OpenAI GPT and return structured feedback. |
| FR8 | System shall allow users to view previous sessions by session ID. |

### Non-Functional Requirements

| **ID** | **Requirement** |
| --- | --- |
| NFR1 | The system should respond with the next question within 3 seconds. |
| NFR2 | The application must support multilingual input and output (≥5 languages). |
| NFR3 | Feedback accuracy should remain above 85% as rated by peer evaluations. |
| NFR4 | System should have 95% uptime when deployed. |
| NFR5 | Resume and feedback data should be stored securely in DynamoDB. |
| NFR6 | Transcription accuracy for audio input should remain above 80%. |
| NFR7 | Feedback generation (GPT) should complete within 6 seconds per session. |

## System Design

### List of Technologies

Hardware:

* User Device: Personal Computer with Microphone (for voice input)
* Server: AWS services (Cloud-hosted APIs and storage)
* Internet Connectivity

Software:

* OS: Ubuntu via WSL (Windows Subsystem for Linux)
* Backend: Python 3.12, Flask (via AWS Chalice for API structure)
* Frontend: Gradio (local interface)
* AI/ML APIs:
  + OpenAI GPT (feedback generation)
  + OpenAI Whisper (transcription)
  + AWS Polly (text-to-speech)
  + AWS Translate (multilingual translation)
* Data Storage: AWS DynamoDB
* Libraries: boto3, gradio, openai, python-docx, pypdf2, librosa, whisper, pydub, dotenv, pytest

### AI Capability Description & Data

AI Capabilities Implemented:

1. **Skill Extraction** – NLP-based extraction of key terms from resume and job description.
2. **Question Matching** – Retrieval of relevant technical/behavioral/situational questions based on extracted skills.
3. **Answer Evaluation** – GPT-based conceptual feedback with references and improvement suggestions.
4. **Voice Transcription** – OpenAI Whisper converts user voice answers into text.
5. **Voice Confidence Analysis** – Uses pitch variation and energy to determine how confident the user sounds.

**Datasets Used:**

* technical\_qna.json – 800 entries with fields: question, answer, type, skills, domain, experience level
* behavioral\_qna.json – 100 entries
* situational\_qna.json – 100 entries  
  (*All datasets are synthetically generated with assistance from Claude, GitHub Copilot, and job portals.*)

### Architecture Design – AI Capability

**A diagram of a diagram

AI-generated content may be incorrect.**

Figure 8: Architecture Design

### Architecture Design – Full Stack

**A chart of a multicolored structure

AI-generated content may be incorrect.**

Figure 9: Full Stack

### Class Diagram

A diagram of a voice interview

AI-generated content may be incorrect.

Figure 10: Class Diagram

### Component Diagram

A diagram of a software flowchart

AI-generated content may be incorrect.

Figure 11: Component Diagram

### Interaction Diagram

A diagram of a process

AI-generated content may be incorrect.

Figure 12: Interaction Diagram 1

Use Case 1: Analyze Input (AI)

A blue rectangle with black text

AI-generated content may be incorrect.

Figure 13: Analyze Input

Use Case 2: Generate Feedback (AI)

A diagram of a diagram

AI-generated content may be incorrect.

Figure 14: Generate Feedback

## Project Plan

A screenshot of a computer

AI-generated content may be incorrect.

Figure 15: Feedback Plan

## Test Plan and Test Cases

The objective of this testing phase was to ensure each module of the AI Interview Bot operates accurately and consistently. The testing included:

* **Unit testing**: Verification of individual modules such as parsing, transcription, feedback generation, translation, and audio confidence scoring.
* **Integration testing**: Ensured proper data flow between components and system-wide functionality.

All tests were written using pytest, and test coverage was recorded using coverage.py.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 16: Test Coverage

| **Module** | **Tested Functions** | **Description** | **Expected Output** |
| --- | --- | --- | --- |
| parser.py | extract\_text() | Test if resume (PDF/DOCX) content is extracted properly | Cleaned string of extracted text |
| skills\_extractor.py | extract\_keywords\_from\_resume\_and\_jd() | Test if skills are being extracted from input text | List of identified keywords |
| question\_retriever.py | get\_final\_question\_set() | Ensures correct question types are selected | 5 questions (3 technical, 1 behavioral, 1 situational) |
| translate.py | translate\_text(), translate\_to\_english() | Validate multilingual translation | Translated output string |
| whisper\_transcriber.py | transcribe\_audio\_local() | Check audio file is correctly transcribed into text | Transcribed string |
| voice\_analyzer.py | analyze\_voice\_confidence() | Analyze pitch and energy from audio | Confidence level string |
| feedback.py | get\_feedback\_from\_gpt() | Sends questions and answers to GPT for structured feedback | Formatted feedback string or dictionary |

Integration Test Plan

| **Scenario** | **Modules Involved** | **Test Description** | **Expected Result** |
| --- | --- | --- | --- |
| Full Interview Workflow (text) | Parser → Extractor → Retriever → Translator → Feedback → DynamoDB | Simulate complete flow with typed answers | Session ID generated and stored |
| Full Interview Workflow (voice) | Transcriber → Analyzer → Translator → Feedback → DynamoDB | Complete interview with audio answers | Feedback includes voice confidence scores |
| Multilingual Input and Feedback | Translator → Feedback → Translator | Translate resume, JD, questions, answers, and feedback | Entire flow works in selected language |
| Session Retrieval | DynamoDB | Retrieve session details from session ID | Correct history displayed |

Test Coverage

| **Module** | **Coverage** |
| --- | --- |
| parser.py | 100% |
| skills\_extractor.py | 100% |
| question\_retriever.py | 95% |
| translate.py | 100% |
| whisper\_transcriber.py | 90% |
| voice\_analyzer.py | 100% |
| feedback.py | 100% |

Total Test Coverage: 98%

## UI/UX Design

The user interface (UI) was designed to be **simple**, **interactive**, and **accessible** to both technical and non-technical users. The experience (UX) focuses on reducing cognitive load during simulated interviews while making advanced features like multilingual input, voice interaction, and real-time feedback feel intuitive.

1. **Landing Tab: “Interview”**
   * Upload Resume (.pdf or .docx)
   * Paste Job Description
   * Choose Interview Language (Dropdown)
   * Click “Start Interview”
2. **Interview Interface (Chatbot)**
   * Bot displays a translated question
   * User can:
     + 🎙️ Record or upload an audio response
     + ✍️ Type an answer manually
   * Bot asks next question until 5 questions are done
   * GPT feedback is displayed in structured format
3. **Feedback**
   * 🔹 Confidence Score (from voice)
   * 💡 Conceptual Feedback (from GPT)
   * 💬 Reference Answers
4. **Session Retrieval**
   * “Session History” tab to view previous interview by session ID

A screenshot of a computer

AI-generated content may be incorrect.

Figure 17: UI/UX

**UX Considerations**

* **Multilingual Flexibility**: UI supports five languages for both input and output, with visual flags for ease of selection.
* **Voice-first Interaction**: Users can comfortably answer using recorded speech, supporting realistic interview simulation.
* **Session Persistence**: All completed interviews are stored with a unique session ID, enabling future review.
* **Audio Feedback**: Polly-generated audio questions and Whisper-based input create a natural two-way voice experience.
* **Minimal Distractions**: UI elements are grouped inside collapsible panels and tabs to reduce screen clutter.

## Deployment Strategy

The goal of the deployment strategy is to outline how the AI Interview Bot can be set up, run, and maintained across development, testing, and production environments. Given that the final version is intended for **local use**, no cloud deployment was performed during this phase. However, the system has been designed to be **easily portable and scalable**, with clear steps for future deployment to cloud platforms like AWS or Azure.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 18: ReadMe file for running code

Deployment Environment

* **Operating System**: Ubuntu (via WSL on Windows)
* **Python Version**: 3.12
* **Virtual Environment**: venv created to isolate dependencies
* **Frontend**: Gradio (local web interface)
* **Backend**: Python-based utility modules (parser, GPT, Whisper, Polly, etc.)
* **Database**: AWS DynamoDB (cloud-based, no local database used)

## High – Level ML/Ops Description

Although the project does not include traditional model training pipelines, it leverages **Machine Learning Operations (MLOps)** principles for managing interactions with pretrained models and external AI services. The system integrates **OpenAI Whisper** for speech-to-text, **AWS Translate** for multilingual handling, and **GPT (via OpenAI API)** for real-time interview feedback.

The architecture is modular, allowing each ML component to operate independently. This promotes **reusability**, **testing**, and **maintainability**, aligning with MLOps best practices.

A diagram of a graph

AI-generated content may be incorrect.

Figure 19: Feedback and Translation Process

| **Component** | **Purpose** | **Type** | **Source** |
| --- | --- | --- | --- |
| OpenAI Whisper | Audio transcription | ML Inference | Local model |
| AWS Translate | Multilingual translation | ML API | AWS |
| AWS Polly | Text-to-speech | ML API | AWS |
| OpenAI GPT (gpt-3.5) | Feedback generation | LLM Inference | OpenAI API |
| Voice Analyzer | Confidence score using spectral analysis | Signal Analysis | Local Python module |

# List of Acronyms

| **Acronym** | **Full Form** |
| --- | --- |
| AI | Artificial Intelligence |
| ML | Machine Learning |
| MLOps | Machine Learning Operations |
| GPT | Generative Pre-trained Transformer |
| TTS | Text-to-Speech |
| NLP | Natural Language Processing |
| AWS | Amazon Web Services |
| API | Application Programming Interface |
| UI/UX | User Interface / User Experience |
| PDF | Portable Document Format |
| DOCX | Microsoft Word Document |
| JSON | JavaScript Object Notation |
| UUID | Universally Unique Identifier |
| WSL | Windows Subsystem for Linux |
| CLI | Command Line Interface |
| CRUD | Create, Read, Update, Delete |
| IDE | Integrated Development Environment |
| DB | Database |
| LLM | Large Language Model |
| ASR | Automatic Speech Recognition |

# Glossary of Terms

| **Term** | **Definition** |
| --- | --- |
| **Artificial Intelligence (AI)** | The simulation of human intelligence processes by machines, especially computer systems, enabling tasks such as learning, reasoning, and self-correction. |
| **Machine Learning (ML)** | A subset of AI focused on enabling machines to learn from data and improve performance without explicit programming. |
| **Large Language Model (LLM)** | An AI model trained on vast amounts of text data to understand and generate human-like language, such as OpenAI’s GPT. |
| **Whisper** | An open-source speech recognition model developed by OpenAI used for transcribing spoken audio into text. |
| **Text-to-Speech (TTS)** | A technology that converts written text into spoken voice output using AI models like AWS Polly. |
| **Confidence Score (Voice)** | An analysis of speech delivery to evaluate clarity, fluency, and hesitancy, used to assess a candidate’s speaking confidence. |
| **Session ID** | A unique identifier assigned to each user interaction or interview session, used for retrieval and record-keeping. |
| **DynamoDB** | A fully managed NoSQL database service by AWS used to store interview session data securely. |
| **Resume Parsing** | The process of extracting structured information (e.g., skills, experience) from a resume document in formats like PDF or DOCX. |
| **GPT Feedback** | Automated feedback generated using a GPT model that provides strengths, areas of improvement, and technical corrections based on user answers. |
| **Translation (AWS Translate)** | A machine learning service by AWS used to translate user input/output into different supported languages. |
| **Gradio** | A Python library used to build interactive user interfaces for ML models and applications in web browsers. |
| **Multilingual Support** | The ability of the system to interact in multiple languages, both in input and output, using translation APIs and language detection. |
| **Session Retrieval** | The functionality allowing users to access previous interview data using a stored session ID. |
| **.env File** | A configuration file used to store environment variables such as API keys securely (excluded from version control). |