AI MOCK INTERVIEW SIMULATOR

A DESIGN PROJECT REPORT

Submitted by

BALAJI S (811722001004)

GOKULRAJ A (811722001010)

RAGHISH S (811722001039)

ROSHAN ABOORVA RAJ J (811722001043)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE, 2025

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this design project report titled **"AI MOCK INTERVIEW SIMULATOR"** is the bonafide work of **BALAJI S (811722001004), GOKULRAJ A (811722001010), RAGHISH S (811722001039), ROSHAN ABOORVA RAJ J (811722001043),** who carried out the design project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other design project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. T.Avudaiappan, M.E., Ph.D.,
HEAD OF THE DEPARTMENT

Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

SIGNATURE

Mrs. Ganga Naidu, M.E .,
SUPERVISOR

ASSISTANT PROFESSOR
Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voce examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the design project report on **"AI MOCK INTERVIEW SIMULATOR"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This design project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

Signature

_____

BALAJI S

_____

GOKULRAJ A

_____

RAGHISH S

_____

ROSHAN ABOORVA RAJ J

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K. Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this design project.

We are glad to credit honourable chairman **Dr. K.RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our design project and offering adequate duration in completing our design project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the design project the full satisfaction.

We whole heartily thank **Dr. T.AVUDAIAPPAN**, **M.E., Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this design project.

We express our deep and sincere gratitude to our design project guide **Mrs. GANGA NAIDU, M.E.,** Department of **ARTIFICIAL INTELLIGENCE,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry out this design project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

The AI Mock Interview Simulator is a voice-interactive desktop application developed to provide users with a realistic, automated environment for practicing interview skills. By integrating multiple advanced technologies, the system offers a seamless and immersive experience that mimics real-life interview scenarios. A graphical user interface built using Tkinter serves as the central control hub, allowing users to initiate and manage interviews with ease. The text-to-speech module, implemented using pyttsx3, delivers interview questions and system prompts audibly, while the speech-to-text module, powered by Vosk and sounddevice, captures and transcribes spoken responses in real time. At the core of the system lies the AI Interview Engine, which utilizes the GPT4All (Mistral) model to dynamically generate relevant interview questions, analyze user responses, and provide detailed feedback and performance scores. An Interview Controller module orchestrates the overall workflow by managing the session state, coordinating input and output across modules, and ensuring a smooth conversational experience. Designed with modularity, usability, and performance in mind, the simulator offers a powerful and efficient tool for developing communication skills, building interview confidence, and receiving personalized, AI-generated insights without requiring human intervention.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligence
HR	-	Human Resources
LLM	-	Large Language Model
NLP	-	Natural Language Processing
NLU	-	Natural Language Understanding
STT	-	Speech-To-Text
TTS	-	Text-To-Speech
UI	-	User Interface

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The AI Mock Interview Simulator is an innovative solution designed to assist students and job seekers in preparing for real-world interviews through a dynamic, voice-enabled virtual environment. This system integrates advanced artificial intelligence, speech-to-text (STT), and text-to-speech (TTS) technologies to deliver an immersive and realistic mock interview experience.

This simulator functions as a conversational agent that assumes the role of an interviewer. Users begin by selecting a specific job role—such as Data Analyst, Software Developer, or Product Manager—and then participate in an interactive interview process. Unlike traditional mock interviews, the interaction occurs entirely through voice communication. The AI interviewer poses questions verbally, and the user responds by speaking. These responses are transcribed in real-time using speech recognition, and the AI evaluates the answers, offers follow-up questions, and ultimately provides a score along with personalized feedback.

This application offers an intuitive and user-friendly interface built with the Tkinter library, ensuring easy access and operation for users. The use of the Vosk model for STT and GPT4All for natural language processing enables intelligent dialogue generation, creating a seamless and engaging mock interview simulation. The system not only mimics real interview scenarios but also records user responses, evaluates them contextually, and offers insights on improvement—helping candidates boost confidence and communication skills before facing actual interviews.

1.2 OBJECTIVE

The primary objective of the AI Mock Interview Simulator is to empower users with a practical, AI-powered interview preparation tool that enhances readiness and performance through simulated role-based interviews. By combining voice interaction, natural language understanding, and intelligent feedback, the system aims to replicate the pressure and spontaneity of real interviews while offering constructive, AI-generated evaluations.

This project is designed to bridge the gap between theory and practice in interview preparation. Users can rehearse responses, receive targeted suggestions, and improve their delivery without the need for human evaluators. The system helps reduce interview anxiety by familiarizing candidates with question formats, improving their articulation, and providing metrics-based performance evaluations.

From a broader perspective, the simulator serves as a personal training assistant that adapts to the chosen job role and supports continuous improvement through speech input analysis and contextual feedback. It promotes self-learning by encouraging users to refine their answers based on AI suggestions. The scoring mechanism, combined with qualitative insights, allows users to measure their growth over multiple sessions.

For educational institutions and training centers, this system can act as a scalable and cost-effective tool to supplement career counseling efforts, especially in scenarios where personalized interview practice may be logistically challenging.

CHAPTER 2

LITERATURE SURVEY

2.1 AN AI MOCK-INTERVIEW PLATFORM FOR INTERVIEW PERFORMANCE ANALYSIS

Yi-Chi Chou, Felicia R. Wongso, Chun-Yen Chao, Han-Yen Yu,2022

This platform integrates artificial intelligence to simulate mock interviews, providing users with real-time performance analysis. It offers a comprehensive suite that includes a resume builder, AI-driven interview simulations, and detailed feedback mechanisms. The system aims to enhance interview preparedness by evaluating user responses, behaviors, and providing actionable insights to refine skills and boost confidence.

Merits

- **Comprehensive Feedback**
  Delivers detailed analysis on user performance, highlighting areas of strength and improvement.
- **Integrated Tools**
  Combines resume building and interview simulation in a single platform, streamlining the preparation process.

Demerits

- **Limited Personalization**
  May not fully adapt to individual user nuances or specific industry requirements.
- **Dependence on AI Accuracy**
  Relies heavily on AI algorithms, which may occasionally misinterpret user responses or behaviors.

2.2 AI CHATBOT FOR JOB INTERVIEW

N. Boudjani; V. Colas; C. Joubert; D. Ben Amor,2023

The AI Chatbot for Job Interview is designed to assist candidates in preparing for job interviews by simulating real-time interview scenarios. Utilizing natural language processing, the chatbot engages users in interactive dialogues, providing instant feedback and suggestions to improve responses. The system aims to enhance user confidence and readiness for actual interviews.

Merits

- **Interactive Practice**
  Offers a conversational interface that mimics real interview dynamics, allowing users to practice effectively.
- **Immediate Feedback**
  Provides real-time suggestions and corrections, facilitating quick learning and improvement.

Demerits

- **Generic Responses**
  May offer feedback that lacks depth or specificity to particular industries or roles.
- **Limited Emotional Intelligence**
  Might not accurately interpret or respond to nuanced human emotions during interactions.

2.3 Q&AI: AN AI POWERED MOCK INTERVIEW BOT FOR ENHANCING THE PERFORMANCE OF ASPIRING PROFESSIONALS

C. J. Joel Manuel,2024

Q&AI is an AI-powered mock interview bot developed to assist aspiring professionals in enhancing their interview performance. The system simulates real interview scenarios, enabling users to practice and receive feedback on their responses. It focuses on improving communication skills, confidence, and overall readiness for job interviews.

Merits

- **Realistic Simulation**
  Emulates actual interview settings, providing users with a practical preparation tool.
- **Skill Enhancement**
  Aims to boost user confidence and communication abilities through repeated practice.

Demerits

- **Limited Customization**
  May not cater to specific industry requirements or personalized user needs.
- **Feedback Depth**
  Feedback provided might lack comprehensive analysis of user performance.

2.4 AI-BASED MOCK INTERVIEW EVALUATOR: AN EMOTION AND CONFIDENCE CLASSIFIER MODEL

Rubi Mandal, Pranav Lohar, Dhiraj Patil, Apurva Patil, Suvarna Wagh,2023

This study presents an AI-based mock interview evaluation system that leverages emotion recognition and confidence classification to assess candidate performance. The model captures and analyzes visual and audio cues such as facial expressions, tone of voice, and speech patterns during simulated interviews. By integrating computer vision and natural language processing, the system quantifies emotional responses and confidence levels, offering structured feedback to help users refine their soft skills and communication under pressure. This technological advancement is aimed at enhancing interview preparedness by offering personalized performance insights.

Merits

- **Emotion-Centric Evaluation**
  Incorporates facial and vocal emotion detection to provide more human-like performance insights.
- **Confidence Analysis**
  Measures user confidence through tone and behavior, helping users understand and improve self-presentation.

Demerits

- **Hardware Dependency**
  Requires access to high-quality audio-visual equipment for accurate analysis, which may limit accessibility.
- **Ethical and Privacy Issues**
  Collecting biometric and emotional data can raise concerns regarding user consent, data security, and ethical usage.

2.5 AI-DRIVEN INTERVIEWER: ENHANCING INTERVIEW EXPERIENCE THROUGH CONVERSATIONAL AI

S. Sumathi, D. Kevin Harris, A. K. Jeyanth,2024

The AI-Driven Interviewer leverages conversational AI to enhance the interview experience for candidates. By integrating voice-enabled chatbots with advanced natural language understanding, the system conducts interactive interviews, providing real-time feedback and personality assessments. It aims to address common challenges faced by job seekers, such as lack of feedback and preparation opportunities.

Merits

- **Conversational Engagement**
  Offers a dynamic and interactive interview simulation, closely mimicking human interaction**.**
- **Personality Insights**
  Provides assessments of personality traits, aiding users in understanding and improving their interview approach.

Demerits

- **Complexity of Implementation**
  The integration of advanced AI technologies may pose challenges in terms of development and maintenance.
- **Potential Bias**
  AI algorithms may inadvertently introduce biases, affecting the fairness of assessments.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In traditional interview preparation methods, students and job seekers typically rely on manual resources such as question banks, peer-to-peer mock interviews, or generalized online tutorials. These approaches often lack interactivity, personalization, and real-time evaluation, resulting in limited feedback and an inconsistent preparation experience. Candidates are often unable to simulate the pressure and dynamics of an actual interview, especially with respect to spontaneous, voice-based communication.

Current systems that attempt to digitize mock interviews are primarily text-based or pre-recorded video simulations. These tools are often static and do not adapt to the user's responses, nor do they offer role-specific questioning or contextual evaluation. Moreover, such systems rarely provide scoring mechanisms or qualitative feedback tailored to the user's spoken content, leaving candidates unaware of their strengths or areas needing improvement.

Voice-based interactions, where available, are usually constrained to specific commands rather than natural language conversations. There is minimal or no use of real-time speech recognition and NLP to assess and adapt based on candidate responses. As a result, users miss the opportunity to refine their speaking skills, clarity, and spontaneity—key aspects evaluated during live interviews.

Additionally, traditional mock interview setups often require the presence of mentors or HR professionals, which may not be scalable or consistently available for all learners. There is also a lack of structured analytics or performance history that candidates can use to track progress over time.

This absence of an intelligent, voice-interactive, self-assessing platform creates a gap between theoretical preparation and the practical skills required in actual interviews.

3.1.1 Demerits

- **Lack of Realistic Simulation**

  Most existing tools fail to provide real-time, dynamic, and voice-based interview experiences that replicate actual interview scenarios.

- **Limited Personalization**

  Generic question sets without role-specific customization do not prepare users for domain-specific interviews.

- **No Speech Analysis or Scoring**

  Current systems lack mechanisms to evaluate spoken responses, provide scores, or offer detailed feedback.

- **Dependence on Human Evaluators**

  Traditional mock interviews often require human involvement, limiting scalability and accessibility.

- **Absence of Performance Tracking**

  Users do not have access to historical performance data or analytics to measure improvement over time.

- **Static and Unengaging User Experience**

  Text-only or prerecorded systems do not engage users in an interactive, conversational manner.

3.2 PROPOSED SYSTEM

The proposed system, AI Mock Interview Simulator, offers a voice-based, AI-driven platform that transforms how students and job seekers prepare for interviews. The simulator is built to provide a realistic, interactive mock interview environment powered by speech recognition, natural language understanding, and intelligent feedback mechanisms.

The system operates through a modular design, beginning with role selection, where users can choose from job profiles like Data Analyst or Web Developer. Once a role is selected, the AI interviewer initiates a verbal interaction using text-to-speech technology. Users respond via microphone, and

the system uses speech-to-text conversion (powered by Vosk) to transcribe responses.

The AI then processes these responses using a local language model (GPT4All), evaluates them based on relevance, coherence, and confidence, and provides a performance score along with detailed feedback. The simulator offers follow-up questions, creating a multi-turn conversation that closely mirrors a real interview scenario.

The graphical interface is developed using Tkinter, offering simplicity and accessibility. The backend handles real-time speech processing, scoring logic, and user interaction flow. This self-contained application is designed to function locally, ensuring privacy and low resource dependency.

By incorporating AI technologies into a compact and efficient design, this system addresses the limitations of traditional preparation methods and offers users a powerful tool to build confidence, improve communication, and enhance interview readiness.

3.2.1 Merits

- **Voice-Based Interaction**

  Provides a realistic simulation through speech input/output, helping users practice spoken communication and build confidence.

- **Role-Specific Questioning**

  Interview questions are tailored to the selected job role, increasing relevance and preparation depth.

- **Automated Evaluation and Scoring**

  The system analyzes answers and gives immediate scores and qualitative feedback to help users improve.

- **Self-Paced and Scalable**

  Can be used independently by any number of users without requiring a live interviewer, making it scalable and accessible.

CHAPTER 4
SYSTEM SPECIFICATIONS

4.1 HARDWARE SPECIFICATIONS

CPU: Dual-core processor or better.
RAM: 4 GB minimum.
Storage: At least 1 GB free space.
Microphone and speakers/headphones.

4.2 SOFTWARE SPECIFICATIONS

Operating System: Windows/Linux/Mac.
Python 3.8+.
Libraries: tkinter, sounddevice, pyttsx3, vosk, gpt4all, numpy, threading.
Vosk Speech Model: vosk-model-en-us-0.22.
LLM Model: mistral-7b-instruct-v0.1.Q4_0.gguf.

4.3 FUNCTIONAL SPECIFICATIONS

Role selection from dropdown (e.g., Software Developer).
Voice-based or typed interaction.
Live interview session with follow-up questions.
Real-time speech-to-text and text-to-speech conversion.
Final evaluation and feedback report.

CHAPTER 5
SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The architecture of the AI Mock Interview Simulator is designed to deliver an interactive, intelligent, and responsive voice-based platform that helps users practice and refine their interview skills in a realistic environment. It is structured into three primary layers: the Presentation Layer, Application Layer, and Data Layer.

The Presentation Layer focuses on the graphical user interface built using Tkinter, providing a simple, intuitive environment for users to interact with the simulator. Users can select their desired interview role (e.g., Web Developer, Data Analyst), initiate the interview, and view real-time feedback and scores. The UI supports voice interactions using microphone input and speaker output, mimicking real interview conditions. The interface is designed for ease of use and can operate on various screen sizes and devices with minimal dependencies.

At the core lies the Application Layer, which contains the system's functional logic and manages the interactions between the user interface, speech modules, and the local AI model. This layer handles:

- Role Selection and Interview Flow: Presents domain-specific questions based on the selected role.
- Voice Interaction: Converts user responses from speech to text using Vosk, and delivers AI-generated questions using text-to-speech (TTS).
- Response Processing: Transcribes and interprets spoken responses, then evaluates them using a local LLM (GPT4All) to generate scores and provide qualitative feedback.
- Conversation Management: Maintains the interview context across multiple questions to ensure coherence.

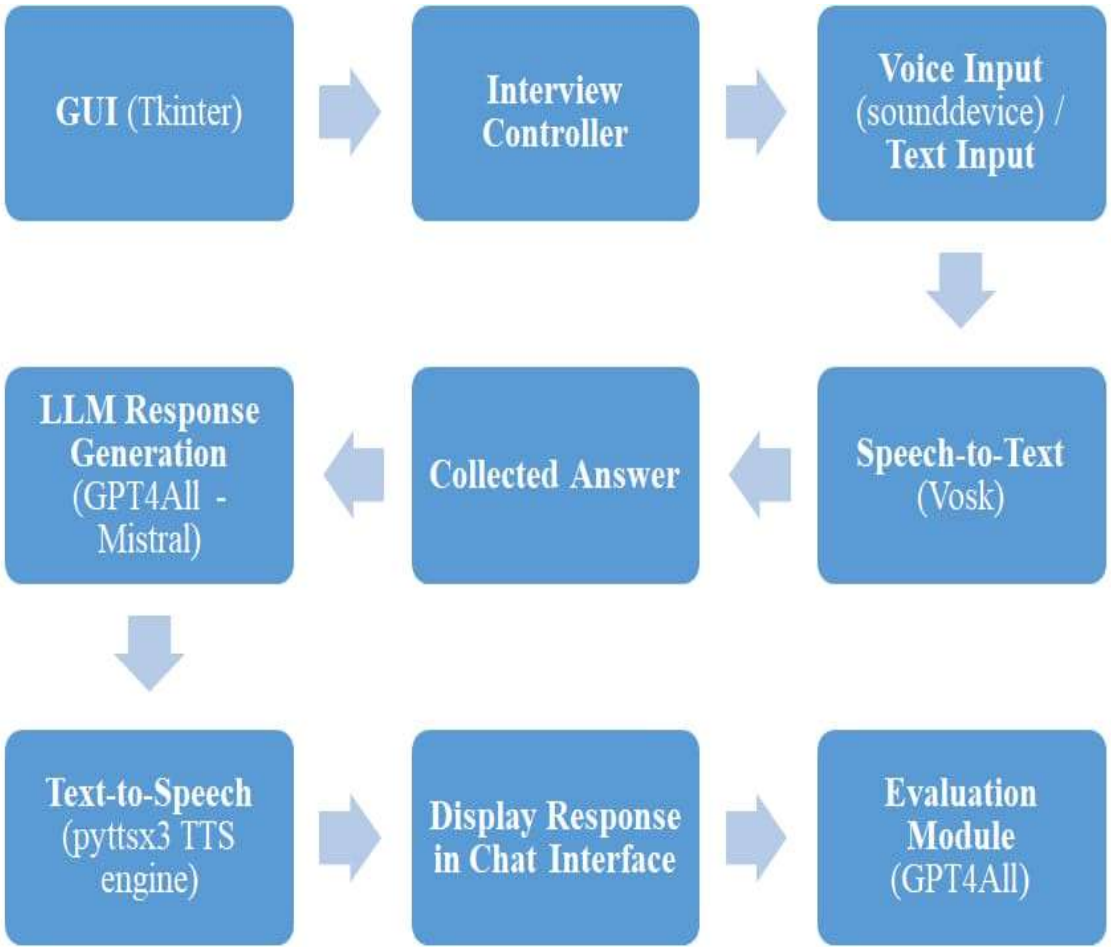

Fig. 5.1 System Architecture

The Data Layer is responsible for handling all data-related tasks, including storing interview questions, logging user responses, maintaining scores, and storing feedback sessions. This layer can use lightweight, local data storage methods such as JSON or SQLite for simplicity and offline functionality. Additionally, system configurations, such as available roles and scoring rubrics, are managed here. The modularity of this layer allows for future integration with cloud databases or advanced analytics dashboards.

To ensure smooth and responsive performance, the entire application runs locally, eliminating dependency on external servers and preserving user privacy. The architecture is designed to be modular and extendable, allowing easy integration of additional features like progress history, advanced analytics, or support for new languages and roles.

This architecture forms the backbone of the AI Mock Interview Simulator, providing a realistic and scalable system that offers personalized, data-driven interview preparation for students and job seekers.

CHAPTER 6
MODULES DESCRIPTION

6.1 USER INTERFACE MODULE

Technology Used: Tkinter

The User Interface (UI) module is the face of the AI Mock Interview Simulator. Built using the Tkinter library in Python, it serves as the medium through which users interact with the various backend functionalities of the system. The UI is crafted to provide an intuitive, responsive, and interactive environment that accommodates users of different proficiency levels, ensuring a smooth experience from start to finish of an interview session.

The interface is designed with usability at the forefront, providing a clean layout where all essential controls are easily accessible. Users are first greeted with a welcome screen or home window featuring key options such as Role Selection, Start Interview, Start/Stop Recording, and Submit Answer. These elements are arranged logically to minimize confusion and enhance the user journey throughout the interview simulation.

The central component of the interface is the chat box, a scrollable text display that reflects the ongoing dialogue between the AI interviewer and the candidate. This chat box dynamically updates with each new question and answer, maintaining the chronological flow of the conversation. Color-coded text or prefixed labels (e.g., “AI:” vs. “You:”) help users quickly distinguish between interviewer messages, user responses, and system-generated notifications.

Additionally, the interface includes a manual input text field, allowing users to enter their responses via the keyboard if they prefer not to use the voice input functionality. This ensures accessibility and inclusivity, catering to users who may face difficulties with voice input or prefer text-based communication.

The UI also handles error messages and system alerts. For instance, if the microphone is not detected or if the system is waiting for the AI to generate the next question, appropriate messages are displayed to keep users informed. These

dynamic prompts contribute to a responsive and adaptive interface that reacts intelligently to user actions and system states.

Furthermore, the modular design of the UI allows for easy integration with other components such as the Speech-to-Text, Text-to-Speech, and AI Interview Engine modules. Each UI action triggers a backend process that ensures synchronization between the graphical interface and the system's core logic.

6.1.1 Role Selector

The Role Selector is an essential feature within the UI, giving users control over the type of interview they wish to simulate. This dropdown menu allows users to choose from predefined roles such as HR Interviewer, Technical Interviewer, or Managerial Interviewer.

Each selected role influences the behavior of the AI Interview Engine, altering the nature of the questions posed during the session. For example, an HR interview might focus on soft skills, behavioral scenarios, and background inquiries, while a technical interview will prioritize programming problems, domain-specific knowledge, or problem-solving questions. This customization feature enhances the realism of the simulation and allows candidates to tailor their practice sessions according to their career aspirations.

The sub-module is event-driven—whenever the user selects a role from the dropdown, it updates an internal variable which is referenced by the AI Interview Engine before generating any question. This ensures that the AI interviewer maintains contextual consistency based on the selected role throughout the interview.

6.1.2 Chat Display

The Chat Display sub-module is a dynamic, scrollable text box embedded within the UI. It serves as the main communication area, displaying real-time conversation between the AI interviewer and the user.

This module is responsible for:

- Maintaining a running log of all exchanged messages.

- Differentiating between system prompts, AI-generated questions, and user responses through color coding, text styling, or labels.
- Automatically scrolling to the newest message, ensuring users always view the latest exchange.
- Preserving context by storing the conversation history within the session, which can be useful for post-interview evaluation or review.
- Moreover, the chat display can be extended to support advanced features like timestamping messages, loading historical data, or exporting the entire session for later review.

6.1.3 Control Panel

The Control Panel sub-module contains the core interactive buttons that manage the interview lifecycle. These include:

- Start Interview: Initializes the AI Interview Engine, preparing it to generate the first question and activating necessary modules like STT and TTS.
- Start/Stop Recording: Controls the activation of the Speech-to-Text module. When pressed, it begins listening to the user's voice; pressing it again will stop the recording and pass the audio for transcription and response handling.
- Submit Answer: Primarily used when the user inputs responses via the text field. This ensures both voice and text input methods are supported.

The Control Panel ensures synchronization between frontend actions and backend logic. For example, pressing the Start Interview button will not only update the UI but also initialize internal interview session variables and timers (if used). It manages state transitions, like switching from idle to recording mode or from one question to the next, maintaining a coherent workflow across the entire system.

Additionally, button states are dynamically enabled or disabled depending on the system's current mode—for instance, disabling the Submit button while recording is active, or disabling Start Interview once the session begins—to prevent conflicting actions and enhance user experience.

6.2 TEXT-TO-SPEECH (TTS) MODULE

Technology Used: pyttsx3

The Text-to-Speech (TTS) module plays a pivotal role in delivering an interactive and immersive interview experience by converting AI-generated text into audible speech. This functionality bridges the gap between machine and human communication, making the simulation more realistic and user-friendly. Leveraging the pyttsx3 library, the system benefits from an offline-capable TTS engine, ensuring uninterrupted service without reliance on external internet-based APIs. This makes the simulator highly reliable and suitable for use in bandwidth-constrained environments such as rural campuses or testing labs.

The primary function of the TTS module is to vocalize questions, system messages, and end-of-session feedback in a clear, professional tone. The output aims to replicate the formal tone of a real-world interviewer, helping users adapt to the stress and pacing of live interviews. This is particularly useful for preparing users for in-person or telephonic interviews where auditory communication is a critical factor.

By providing audible prompts instead of relying solely on text-based interfaces, the TTS module also enhances accessibility for users with reading or visual impairments. Additionally, it introduces a sense of engagement, particularly helpful for candidates who want to practice listening and responding as they would in real scenarios. The system can announce the beginning of the interview, deliver each question, prompt the user to speak or submit their response, and finally present a synthesized verbal summary of their performance.

The TTS module operates in tight integration with the Interview Controller, which determines when the system needs to speak. This ensures that each phase of the interview process—starting, questioning, listening,

evaluating—is accompanied by appropriate voice prompts, allowing users to focus fully on the interaction without the need to constantly check the screen.

To provide a user-friendly and natural auditory output, the TTS engine is customizable. It supports a range of configuration options like speaking rate (words per minute), volume, and voice selection (male or female). These settings can be tailored to suit different users' preferences or to simulate different interviewer personas, enhancing the realism and adaptability of the simulator.

6.2.1 Speech Synthesis

The Speech Synthesis Sub-module is the core component responsible for converting textual data into spoken audio. When the Interview Controller issues a command to prompt the next question, this sub-module receives the input text string and processes it through the pyttsx3 engine. It then generates a corresponding voice output in real time.

This sub-module is designed to operate efficiently and without noticeable delay, ensuring that there is no awkward silence between interactions. The speech synthesis engine operates asynchronously, meaning it can begin playback even as new text is being prepared in the background. This responsiveness adds to the natural feel of the interaction and keeps users engaged throughout the session.

Additionally, the sub-module supports speaking dynamic strings, allowing it to vocalize variable content such as the user's name, role-specific questions, custom messages, and evaluation metrics. For example, at the end of the interview, it might say, "Your performance score is 8 out of 10. You demonstrated strong technical skills but need to improve your confidence when answering behavioral questions."

This sub-module also integrates logging and error handling mechanisms. In case of any failure in voice synthesis—such as if the audio device is unavailable—fallback messages or visual alerts are triggered to inform the user without disrupting the session.

6.2.2 Voice Configuration

The Voice Configuration Sub-module allows dynamic customization of various voice parameters to optimize user experience. This includes:

- Speech Rate: Controls the speed at which text is spoken. A faster rate can simulate a fast-talking interviewer, while a slower pace might be useful for new users or those practicing in a second language.
- Voice Gender and Accent: pyttsx3 allows switching between male and female voices, depending on the available voices installed on the system. Users or developers can tailor this to match the role selected (e.g., assigning a deeper male voice to a managerial interviewer or a softer female voice for HR).
- Volume Control: This parameter adjusts how loudly the voice is played. Useful for ensuring clarity in different ambient noise conditions.
- Pitch and Emphasis (future extensibility): Although not fully customizable via pyttsx3 in all systems, future expansions can include emphasis or emotional tone modulation for more realistic and expressive speech output.

  Voice configuration not only enhances immersion but also improves inclusivity by letting users select the voice profile they are most comfortable with. This sub-module is also essential for potential future development, such as simulating interviews in different languages or adding multi-accent capabilities for multinational company mockups.

In conclusion, the Text-to-Speech Module is a critical part of the system that transforms a basic Q&A tool into a sophisticated, engaging, and accessible simulation platform. By emulating the auditory dynamics of a real interview, it fosters an environment where users can develop their active listening, reaction timing, and verbal communication skills—key competencies in real-life interview scenarios.

6.3 SPEECH-TO-TEXT (STT) MODULE

Technology Used: Vosk, sounddevice

The Speech-to-Text (STT) Module is an essential component that bridges human interaction and machine interpretation by transcribing spoken language into textual format. This allows the AI Mock Interview Simulator to accept voice responses from the user, process them into machine-readable text, and evaluate or generate relevant follow-up prompts based on those responses. The use of Vosk, an open-source, offline-capable speech recognition toolkit, along with the sounddevice library for real-time audio input, ensures the system functions reliably without internet connectivity, making it especially well-suited for standalone or low-resource environments.

When the user begins speaking after clicking the "Start Recording" button, the STT module activates a continuous audio capture stream through the system's default microphone. The sounddevice library handles real-time audio buffering and routing, while the Vosk engine transcribes the buffered input using an acoustic model and a language model. These components work in tandem to recognize words, segment them into meaningful phrases, and deliver the most accurate transcription possible.

The module supports both interim (real-time) and final transcription results. Interim results are continuously updated as the user speaks, while final results are determined once a pause or sentence boundary is detected. Only finalized transcriptions are forwarded to the AI Interview Engine for evaluation or further questioning, thereby ensuring that partial, ambiguous, or interrupted utterances are not misinterpreted.

This STT system significantly enhances user engagement by enabling natural verbal communication, closely simulating a real-life interview environment. It trains users to develop fluency, clarity, and presence during speech—all vital components of effective interviewing. Furthermore, the STT module is engineered to filter out background noise, detect awkward silences or extended pauses, and respond appropriately when the microphone fails or audio input becomes unintelligible.

Its tight integration with the Interview Controller Module ensures seamless transitions between speaking and listening phases of the mock interview. This results in a smooth, real-time conversation flow between the user and the AI interviewer, minimizing user frustration and maximizing usability.

6.3.1 Audio Capture

The Audio Capture Sub-module is responsible for initiating and maintaining the live audio stream from the system's microphone. Using the sounddevice library, it ensures audio is sampled at a consistent rate (typically 16 kHz or 44.1 kHz), providing a balance between quality and processing efficiency.

This sub-module starts recording when triggered by the "Start Recording" button in the User Interface. It uses asynchronous callbacks to capture chunks of audio data and feed them to the Vosk recognizer in real-time. The sub-module includes buffering mechanisms to prevent data loss and handles interruptions like microphone disconnections or application minimization.

To ensure that the captured audio is clean and conducive for accurate transcription, the sub-module includes preprocessing logic to reduce white noise and eliminate feedback or echo. It also integrates basic volume thresholding to ignore minor ambient sounds that may trigger false positives during transcription.

The modular nature of the audio capture system allows it to be extended or replaced easily with more advanced solutions in the future, such as multi-microphone array processing or noise-cancellation engines.

6.3.2 Real-Time Transcription

At the heart of the STT module is the Real-Time Transcription Sub-module, which takes the live audio feed from the Audio Capture Sub-module and processes it into text using the Vosk speech recognition engine. Vosk models support multiple languages and can be tailored for specific vocabulary domains—such as technical, HR, or behavioral terms—improving accuracy in role-specific interviews.

As audio is streamed, Vosk performs real-time analysis to detect phonemes, match them against a trained language model, and convert them into readable strings. The sub-module handles two key outputs:

- Partial Results: Displayed dynamically while the user is still speaking. These allow a real-time view of the user's progress but are not sent for evaluation.
- Final Results: Determined once a brief silence or pause indicates sentence completion. These are used for further analysis and displayed in the UI's chatbox.

This sub-module includes logic for punctuation restoration and sentence formatting, providing clean, readable input for the AI Interview Engine. Additionally, it tracks confidence scores to assess the reliability of the transcribed content and may discard or flag results that fall below a certain threshold.

This sub-module supports context-aware transcription, which can be refined with additional domain-specific models to handle jargon, acronyms, or frequently used phrases within a specific industry.

6.3.3 Error Handling and Timeout

The Error Handling and Timeout Sub-module is designed to ensure the robustness and reliability of the STT system, even under imperfect conditions. It actively monitors for anomalies such as:

- Extended Silences: If the user stops speaking for a defined period (e.g., 5 seconds), the system assumes the response is complete and ends the transcription session automatically.
- Microphone Failures: If the microphone becomes inaccessible or the device is unplugged, the sub-module raises an alert and prompts the user with visual or audio cues to reconnect their audio input.
- Unintelligible Input: In cases where the transcription confidence is too low or the input is mostly noise, the system displays a message like "Sorry, we couldn't understand you. Please try again."

Timeouts ensure that each interview question is responded to in a timely manner, helping mimic the real-world pressure of live interviews. These mechanisms also prevent the AI Interview Engine from hanging or misbehaving due to unexpected input delays or hardware issues.

As a part of its feedback system, the sub-module can display user-facing messages such as “Please speak clearly” or “We detected silence, moving to the next question,” keeping the user informed and reducing anxiety during usage.

In sum, the Speech-to-Text Module is a critical enabler of natural interaction in the AI Mock Interview Simulator. It translates spontaneous verbal responses into structured text, facilitating realistic simulations and accurate assessments. By incorporating robust audio handling, real-time transcription, and intelligent error management, this module ensures the system performs effectively across a range of user environments and speech styles. It empowers users to practice their articulation, pacing, and verbal confidence—skills that are indispensable for success in real-life interview situations.

6.4 AI INTERVIEW ENGINE

Technology Used: GPT4All (Mistral model)

The AI Interview Engine serves as the central intelligence of the AI Mock Interview Simulator. Powered by the Mistral variant of GPT4All, this offline large language model (LLM) is designed to emulate the behavior of a human interviewer by analyzing candidate responses, formulating context-aware follow-up questions, and ultimately evaluating performance through qualitative and quantitative metrics.

This module is tasked with simulating realistic, adaptive conversations that not only test a candidate's technical and behavioral knowledge but also mimic the subtle progression of real-world interviews. It dynamically adjusts its tone, questioning depth, and topic flow based on the user’s chosen role (e.g., HR, Technical, Managerial), thus offering a tailored experience for every session.

At the start of the interview, the engine is initialized with a specific persona, which determines its linguistic style, question complexity, and focus

areas. For example, a technical interviewer persona may prioritize problem-solving and domain-specific terminology, while an HR persona may explore soft skills, values, and attitude. The AI Interview Engine sustains context through a structured conversation history, ensuring logical continuity between questions and responses across multiple turns.

The strength of this module lies in its ability to adapt. It doesn't simply present static questions; instead, it interprets previous answers to drive the conversation forward, often probing deeper into topics that a user seems confident or uncertain about. At the end of the session, the engine consolidates all user responses and performs a thorough evaluation based on multiple criteria such as clarity, confidence, relevance, fluency, and coherence. This feedback, along with a scoring rubric, is presented to the user in a structured report, aiding self-improvement and iterative practice.

6.4.1 Question Generation

The Question Generation Sub-module is responsible for producing contextually relevant and logically sequenced questions throughout the interview. It implements the chat() function, which interfaces with the GPT4All LLM to generate content based on the user's selected role and ongoing conversation history.

When initialized, the model is primed with a prompt template tailored for the chosen role (e.g., “You are a Senior Software Engineer interviewing a backend developer.”). This ensures that the generated questions reflect the language and logic typical of that domain. The AI draws on prior user answers to determine which areas require further exploration, generating follow-up questions that delve deeper into the candidate’s competencies or address previously mentioned topics.

For example:

- If a candidate mentions "REST APIs" in a previous answer, the next question might be, "Can you explain how REST differs from GraphQL?"

- In an HR setting, a statement like "I enjoy working collaboratively" might prompt, "Can you describe a time when collaboration led to a successful outcome?"

  This sub-module ensures that questions:
- Vary in difficulty based on earlier responses.
- Maintain a professional and interviewer-like tone.
- Avoid repetition by referencing the Context Management Sub-module.
- Adhere to the session's overall flow and objectives.

  The question generation logic also allows for modular enhancements. Developers may integrate domain-specific question banks or add randomness to increase variety in practice interviews.

6.4.2 Answer Evaluation

The Answer Evaluation Sub-module is activated after the interview concludes, analyzing all of the user's responses to provide a comprehensive assessment. It utilizes the evaluate_answers() function, which applies natural language understanding (NLU) techniques via the GPT4All model to judge each response on multiple dimensions:

- Clarity: Is the answer logically structured and easy to understand?
- Relevance: Does the response directly address the question asked?
- Confidence: Does the language used reflect certainty and command over the topic?
- Depth: Does the candidate elaborate on ideas with appropriate technical or behavioral insight?

  This sub-module generates both qualitative feedback (e.g., “Your answers were clear but could benefit from more technical depth.”) and quantitative scoring, usually on a scale (e.g., 1 to 10) for each category.

  The final output is a well-formatted performance report, which may include:
- Category-wise score breakdown
- Summary comments from the “AI interviewer”
- Suggestions for improvement

- A cumulative score representing overall performance

  This feedback mechanism plays a pivotal role in helping candidates identify strengths and weaknesses. It supports repeat practice by showing measurable progress over time and encourages users to refine both content and delivery of their answers.

6.4.3 Context Management

The Context Management Sub-module ensures that the AI maintains awareness of the interview's progress and adapts accordingly. It maintains a structured record of all previous interactions—questions asked, user responses, and any follow-up context—to enable multi-turn coherence in conversation.

In GPT-based interactions, contextual continuity is vital. Without this module, the AI might generate disjointed or repetitive questions. This sub-module prevents such issues by:

- Logging all exchanges and updating the prompt history after every turn.
- Highlighting important user input keywords or phrases to guide future questioning.
- Filtering out redundant questions and ensuring smooth topic transitions.

  For instance, if the user has already answered a question about their greatest strength, the engine avoids repeating or rephrasing similar prompts. Instead, it may pivot to an adjacent topic like how that strength was applied in a project.

This sub-module also ensures alignment between the Role Selector Sub-module in the User Interface and the personality of the AI interviewer. It persists role-specific parameters across all turns, ensuring tone, terminology, and questioning style remain consistent.

As interviews grow longer and more complex, the need for memory-efficient and relevant context grows. This sub-module may be enhanced in the future to include:

- Session summarization capabilities
- Keyword tagging for rapid response referencing

- Integration with user profiles for long-term progress tracking

The AI Interview Engine forms the cognitive backbone of the AI Mock Interview Simulator. By integrating dynamic question generation, intelligent response evaluation, and robust context management, this module offers users an experience that closely mimics human-led interviews. It adapts in real time to user inputs, maintains logical conversation flow, and provides actionable feedback that supports continuous learning. With the power of the GPT4All Mistral model, this engine operates entirely offline, making it both accessible and versatile across various environments and user scenarios.

6.5 INTERVIEW CONTROLLER

Technology Used: Core Python

The Interview Controller module functions as the brain and traffic coordinator of the AI Mock Interview Simulator. Acting as the central mediator among all functional components—including the User Interface (UI), Text-to-Speech (TTS), Speech-to-Text (STT), and the AI Interview Engine—it ensures a seamless, well-synchronized user experience from the beginning of the mock interview to the final evaluation.

This module encapsulates the full life cycle of an interview session. When the user initiates the session by pressing the "Start Interview" button on the GUI, the controller begins its orchestration by initializing all necessary states and loading the selected interviewer role. It immediately triggers the AI Interview Engine to produce the first question, which is then handed over to the TTS module for vocalization. Once the question has been asked, the controller enters a passive listening state, waiting for the user's response via either the typing interface or voice input.

The moment a user provides input—whether by speaking into the microphone or entering text manually—the controller routes this input to the appropriate handler: if it's audio, it is transcribed via the STT module; if it's text, it is passed directly to the AI engine after some preprocessing. After every

response, the controller commands the AI Interview Engine to generate the next question based on the updated context. This cycle continues until a preset question limit is reached or the user ends the interview session manually.

At the end of the session, the controller triggers the evaluation process, calling upon the AI Interview Engine's scoring and feedback mechanism. It then ensures that the results are shown in the chat interface and spoken aloud via the TTS module. This full-cycle coordination—starting, maintaining, and terminating an interview session with all essential subsystems—makes the Interview Controller the most mission-critical component of the application.

6.5.1 State Management

The State Management Sub-module is responsible for tracking and controlling the internal states of the application. It ensures that the right operations occur in the correct sequence and prevents any component from functioning out of order or under incorrect conditions.

The primary states include:

- Idle: The default state when the app is launched or reset.
- Interview in Progress: Activated once the user starts the interview. Enables voice/text input and question generation.
- Recording: Indicates that the system is actively capturing audio.
- Paused: If the user temporarily halts the session (optional extension).
- Completed: Marks the conclusion of the interview and enables result generation.

This sub-module ensures smooth transitions between these states and enforces rules that prevent invalid actions—for example, it prevents the user from recording a response before the question has been spoken or from starting a new interview without first completing the current one.

Key functionalities:

- Maintains Boolean flags for each major activity (interview_active, recording_enabled).

- Provides an internal state machine or condition checker that synchronizes input/output flows.
- Triggers alerts or GUI prompts when unexpected actions occur (e.g., trying to record without starting the interview).

6.5.2 Input Aggregator

The Input Aggregator Sub-module serves as the central input handler. It collects user responses from both the typing field and the speech transcription engine, processes them, and prepares them for submission to the AI Interview Engine.

Key responsibilities include:

- Multimodal Input Handling: Accepts inputs from different formats—typed text or transcribed audio—and unifies them under a common data structure.
- Sanitization: Cleans up user responses by trimming whitespace, filtering out filler words (optional), and checking for empty or invalid inputs.
- Time Stamping (optional): Attaches timestamps to responses for potential analysis or feedback on response time.
- Role-Specific Parsing (optional): Parses technical inputs to highlight code snippets or terminology if the selected role is technical.

This sub-module ensures that the AI Engine receives only relevant and cleaned data. For voice responses, it waits until the transcription confidence surpasses a predefined threshold or until silence is detected, signifying the end of the response. This avoids sending partial or unclear answers for evaluation or follow-up questioning.

6.5.3 Evaluation Trigger

The Evaluation Trigger Sub-module is responsible for initiating and handling the post-interview evaluation phase. Once the interview concludes—either by reaching a predefined number of questions or through user

termination—the controller calls upon the evaluate_answers() function from the AI Interview Engine to assess user performance.

Key processes include:

- Session Completion Detection: Listens for the session endpoint, which could be triggered automatically (e.g., after 5 questions) or manually (e.g., by clicking “End Interview”).
- Evaluation Call: Sends the full conversation history or only the user’s answers to the AI model for analysis.
- Result Handling: Collects feedback and scores returned by the AI and formats them for both visual and auditory delivery.

The sub-module then updates the chat interface with a final message such as “Your interview is complete. Here is your performance report,” and uses the TTS module to speak out a summary of the results. Additionally, it may offer a prompt or button to allow the user to restart the interview, save the report, or exit the application.

This sub-module ensures closure and a clear endpoint to every session. It’s vital for providing the user with actionable insights and making the simulation feel complete and reflective.

The Interview Controller ties together every functional layer of the AI Mock Interview Simulator, transforming independent modules into a cohesive and intelligent system. Its responsibilities—ranging from session state management to intelligent input aggregation and performance evaluation—make it indispensable for delivering a seamless and responsive interview simulation. Designed with modularity and extensibility in mind, this controller allows future integration with cloud storage, advanced analytics, and cross-platform GUI frameworks without significant architectural overhaul.

CHAPTER 7
CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The AI Mock Interview Simulator represents a significant step forward in preparing job seekers and students for real-world interviews by leveraging artificial intelligence. The system offers an interactive and realistic interview experience through voice-based AI agents, simulating diverse roles and interview scenarios.Users benefit from personalized interviews that adapt to their selected roles, providing dynamic questions, real-time evaluation of responses, and constructive feedback to improve performance. This immersive environment helps users build confidence, enhance communication skills, and identify areas for improvement before facing actual interviews.

7.2 FUTURE ENHANCEMENT

While the AI Mock Interview Simulator provides a robust foundation for interactive interview practice, several enhancements can further improve its capabilities and user experience.

One promising future enhancement is the integration of natural language understanding (NLU) and sentiment analysis. This would allow the system to better interpret the emotional tone, confidence level, and clarity of the user's responses, enabling more nuanced feedback beyond just content accuracy.Another potential improvement is to incorporate multimodal interaction, such as facial expression and body language analysis via webcam, to simulate in-person interview dynamics more closely. This would offer users holistic feedback on their communication skills.

APPENDIX A
SOURCE CODE

```
import tkinter as tk
from tkinter import scrolledtext, ttk
import threading
import sounddevice as sd
import queue
import sys
import pyttsx3
import numpy as np
from vosk import Model, KaldiRecognizer
from gpt4all import GPT4All
tts = pyttsx3.init()
def speak(text):
    tts.say(text)
    tts.runAndWait()
q = queue.Queue()
samplerate = 16000
vosk_model = Model("vosk-model-en-us-0.22")
rec = KaldiRecognizer(vosk_model, samplerate)
recording = False
user_answers = []  # Collect user answers for evaluation later
def callback(indata, frames, time, status):
    if status:
        print(status, file=sys.stderr)
    q.put(bytes(indata))
def listen_until_stop():
    final_text = ""
    try:
```

```
        with sd.RawInputStream(samplerate=samplerate, blocksize=8000,
dtype='int16',
                               channels=1, callback=callback):
            while recording:
                if not q.empty():
                    data = q.get()
                    if rec.AcceptWaveform(data):
                        result = rec.Result()
                        text = eval(result).get("text", "")
                        print(f"Interim text: {text}")
                        if text:
                            final_text += " " + text
            partial_result = rec.FinalResult()
            partial_text = eval(partial_result).get("text", "")
            if partial_text:
                print(f"Final text: {partial_text}")
                final_text += " " + partial_text
    except Exception as e:
        print(f"Error during listen: {e}")
    return final_text.strip()
llm = GPT4All("mistral-7b-instruct-v0.1.Q4_0.gguf", model_path="model/")
def chat(user_answer, current_question, role):
    prompt = (
        f"You are a mock interviewer for a {role} role.\n"
        f"The candidate says: '{user_answer}'\n"
        f"Give a natural, conversational, short follow-up or next question."
    )
    return llm.generate(prompt, max_tokens=80).strip()
def evaluate_answers(answers, role):
    answers_text = "\n".join(f"Answer {i+1}: {ans}" for i, ans in
enumerate(answers))
```

```
        prompt = (
            f"You are a professional interviewer evaluating a mock interview for a {role} position.\n"
            f"Here are the candidate's answers:\n{answers_text}\n"
            f"Give a final evaluation score out of 10 with 1-2 lines of feedback."
        )
        return llm.generate(prompt, max_tokens=100).strip()
    class InterviewApp:
        def __init__(self, root):
            self.root = root
            self.root.title("Mock Interview AI")
            self.root.geometry("800x700")
            self.root.configure(bg="#121212")
            self.interview_active = False
            self.current_question = "Can you please introduce yourself?"
            self.selected_role = tk.StringVar(value="Data Analyst")
            title = tk.Label(root, text="👨‍💼 Mock Interview AI", font=("Helvetica", 20, "bold"), bg="#121212", fg="#ffffff")
            title.pack(pady=10)
            role_frame = tk.Frame(root, bg="#121212")
            role_frame.pack(pady=5)
            tk.Label(role_frame, text="Select Role:", bg="#121212", fg="white", font=("Helvetica", 12)).pack(side=tk.LEFT)
            self.role_menu = ttk.Combobox(role_frame, textvariable=self.selected_role, values=[
                "Data Analyst", "Software Developer", "Product Manager", "Marketing Executive"
            ], font=("Helvetica", 12), state="readonly", width=30)
            self.role_menu.pack(side=tk.LEFT, padx=10)
            self.chat_box = scrolledtext.ScrolledText(root, wrap=tk.WORD, font=("Helvetica", 12), state="disabled",
```

```
                                bg="#1e1e1e", fg="#ffffff",
insertbackground="white", padx=10, pady=10)
        self.chat_box.pack(padx=20, pady=10, fill=tk.BOTH, expand=True)
        input_frame = tk.Frame(root, bg="#121212")
        input_frame.pack(pady=5, fill=tk.X)
        self.entry = tk.Entry(input_frame, font=("Helvetica", 12), bg="#1e1e1e",
fg="white", insertbackground="white")
        self.entry.pack(side=tk.LEFT, padx=10, pady=5, fill=tk.X, expand=True)

        self.send_btn = tk.Button(input_frame, text="Send",
command=self.send_text, font=("Helvetica", 12),
                                  bg="#ff0050", fg="white", activebackground="#e60045",
width=8)
        self.send_btn.pack(side=tk.RIGHT, padx=10)
        controls = tk.Frame(root, bg="#121212")
        controls.pack(pady=10)
        self.start_btn = tk.Button(controls, text="🎤 Start Interview",
command=self.toggle_interview,
                                   font=("Helvetica", 12), bg="#4CAF50", fg="white",
activebackground="#43a047", width=18)
        self.start_btn.grid(row=0, column=0, padx=10)
        self.rec_btn = tk.Button(controls, text="🎙 Start Recording",
command=self.start_recording,
                                 font=("Helvetica", 12), bg="#ff9800", fg="white",
activebackground="#fb8c00", width=18)
        self.rec_btn.grid(row=0, column=1, padx=10)
        self.stop_rec_btn = tk.Button(controls, text="⏹ Stop Recording",
command=self.stop_recording,
                                      font=("Helvetica", 12), bg="#f44336", fg="white",
activebackground="#e53935", width=18)
```

```
        self.stop_rec_btn.grid(row=0, column=2, padx=10)
        self.say_and_display("Interviewer", self.current_question)
    def say_and_display(self, speaker, message):
        self.chat_box.configure(state="normal")
        self.chat_box.insert(tk.END, f"{speaker}: {message}\n\n")
        self.chat_box.configure(state="disabled")
        self.chat_box.yview(tk.END)
        speak(message)
    def toggle_interview(self):
        global user_answers
        self.interview_active = not self.interview_active
        if self.interview_active:
            user_answers = []
            self.start_btn.config(text="■ Stop Interview", bg="#f44336")
            self.say_and_display("Interviewer", "Let's begin the mock interview.")
        else:
            self.start_btn.config(text=" 🎤 Start Interview", bg="#4CAF50")
            self.say_and_display("Interviewer", "Interview stopped. Thank you!")
            role = self.selected_role.get()
            score = evaluate_answers(user_answers, role)
            self.say_and_display("Evaluation", score)
    def send_text(self):
        if not self.interview_active:
            return
        user_input = self.entry.get().strip()
        if not user_input:
            return
        self.entry.delete(0, tk.END)
        self.say_and_display("You", user_input)
        user_answers.append(user_input)
        threading.Thread(target=self.respond, args=(user_input,)).start()
```

```
    def respond(self, user_input):
        role = self.selected_role.get()
        response = chat(user_input, self.current_question, role)
        self.current_question = response
        self.say_and_display("Interviewer", response)
    def start_recording(self):
        global recording
        if not self.interview_active:
            return
        recording = True
        self.say_and_display("System", "Recording started. Speak now...")
        self.listen_thread = threading.Thread(target=self.listen_and_process)
        self.listen_thread.start()
    def stop_recording(self):
        global recording
        if not self.interview_active:
            return
        recording = False
        self.say_and_display("System", "Recording stopped. Processing...")
        self.root.after(500, lambda: print("Allowing buffer to process..."))
    def listen_and_process(self):
        text = listen_until_stop()
        if text:
            self.say_and_display("You", text)
            user_answers.append(text)
            self.respond(text)
if __name__ == "__main__":
    root = tk.Tk()
    app = InterviewApp(root)
    speak("Welcome to Mock Interview AI. You can type or speak your answers.")
    root.mainloop()
```

APPENDIX B

SCREENSHOTS

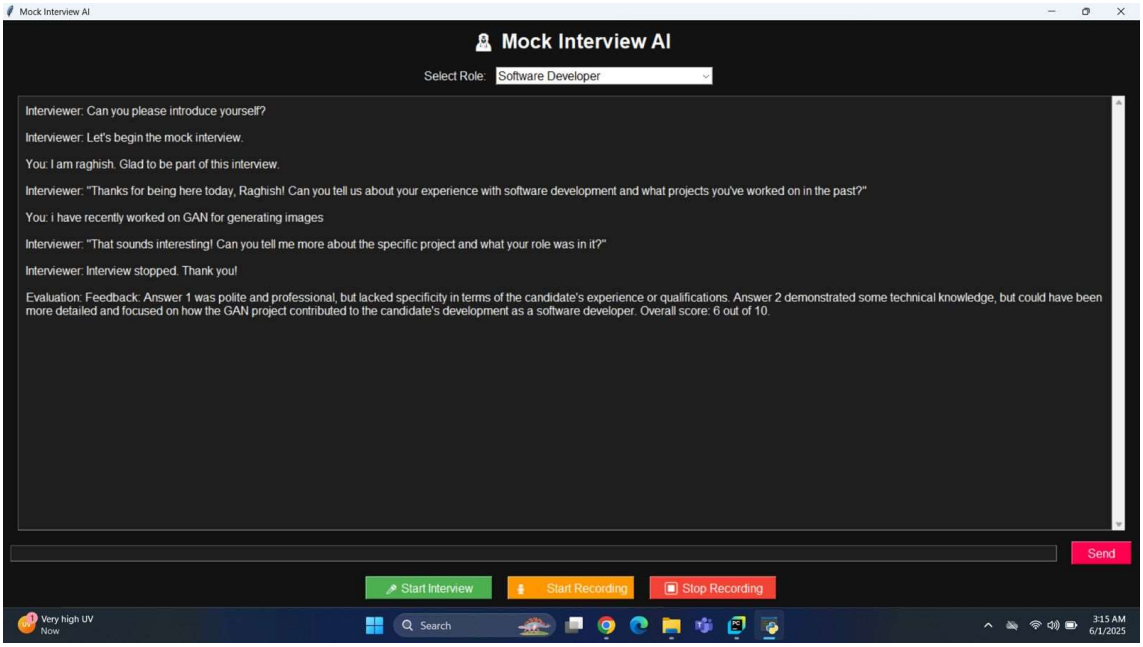

Fig.B.1 Dash Board

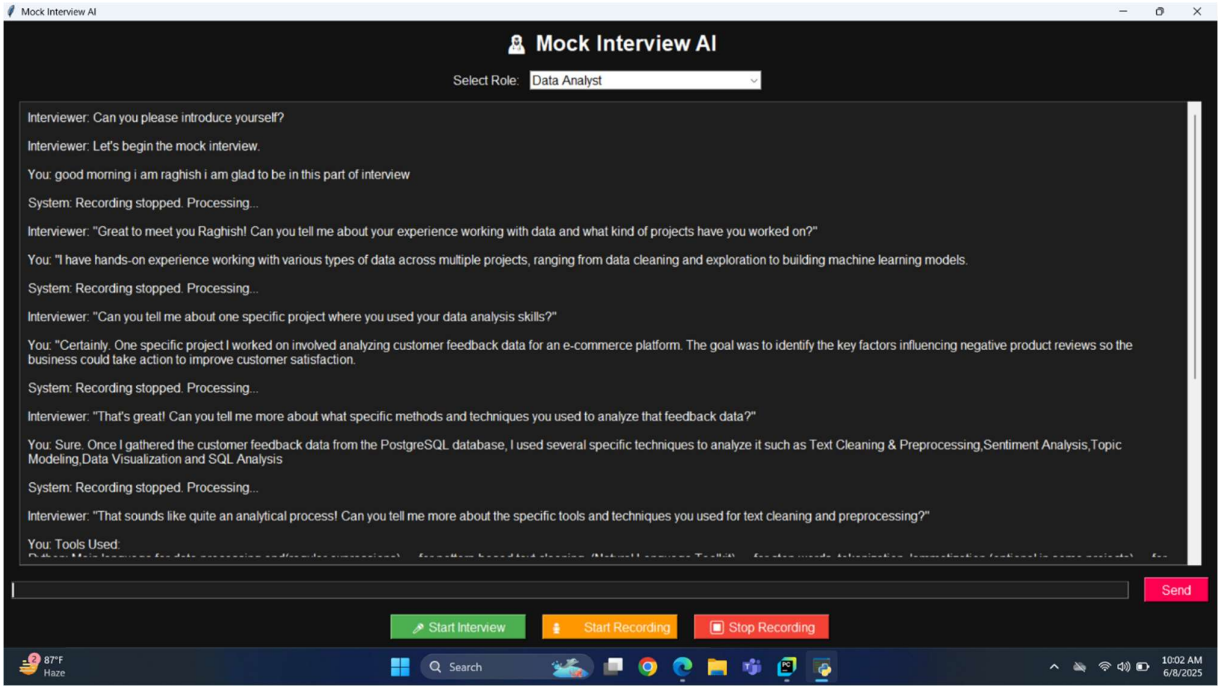

Fig.B.2 Interview Process

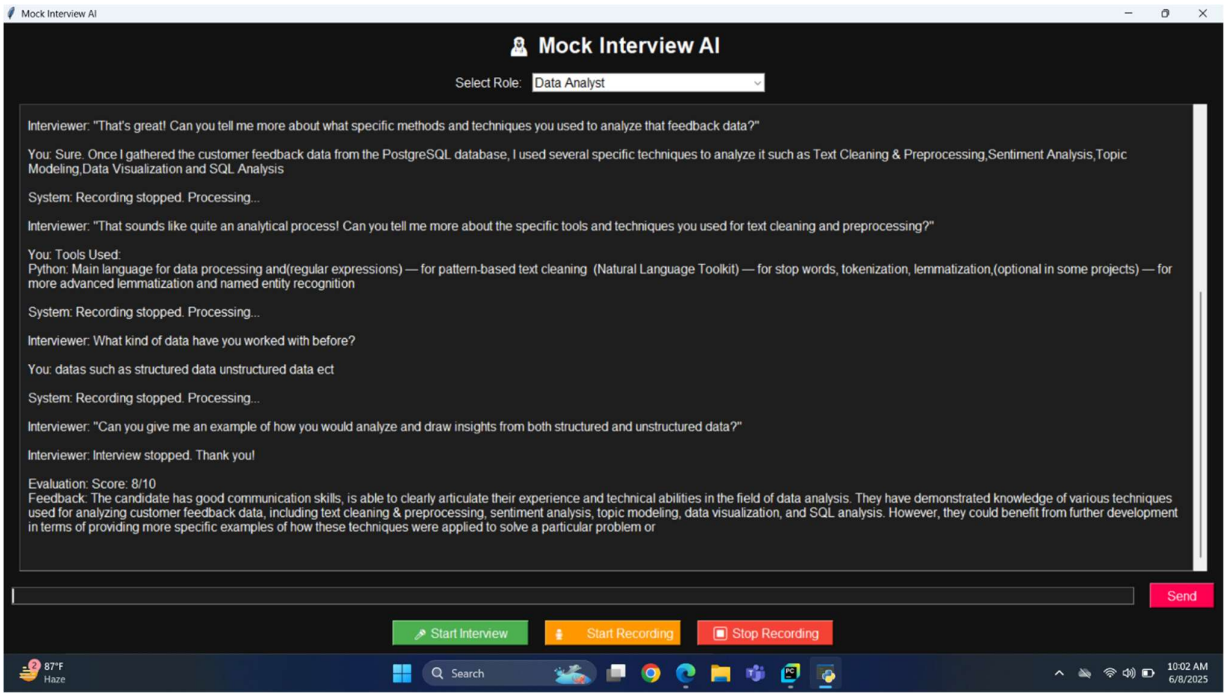

Fig.B.3 Feedback And Score

REFERENCES

1. S. Barari, J. Angbazo, N. Wang, L. M. Christian, E. Dean, Z. Slowinski, and B. Sepulvado, "AI-Assisted Conversational Interviewing: Effects on Data Quality and User Experience," arXiv preprint arXiv:2504.13908, 2025.
2. Z. Liu, "Interview AI-ssistant: Designing for Real-Time Human-AI Collaboration in Interview Preparation and Execution," arXiv preprint arXiv:2504.13847, 2025.
3. N. S. Rai, A. K. R., A. P., and H. N. R., "AI Based Interview Evaluator: An Emotion and Confidence Classifier," International Journal of Advanced Research in Computer and Communication Engineering, vol. 13, no. 4, pp. 1–5, 2025.
4. Wang. R, Patel. S. “Speech and Sentiment Analysis in AI Interviews”,2023.
5. Mandai, Rubi, Lohar, Pranav, Patil, Dhiraj, Patil, Apurva, Wagh, Suvarna. “AI-Based Mock Interview Evaluator”: An Emotion and Confidence Classifier. 2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS), 2023.
6. Chou, Yi-Chi, Wongso, Felicia R., Chao, Chun-Yen, Yu, Han-Yen. “An AI Mock-Interview Platform for Interview Performance Analysis”. Proceedings of the 2022 10th International Conference on Information and Education Technology (ICIET), 2022.
7. S. Dharmatti, A. S. Patil, and R. R. Patil, "Interview Practice-Voice-Based Chatbot," International Journal of Advances in Engineering Research, vol. 23, pp. 45–50, 2022.
8. X. Han, M. Zhou, M. Turner, and T. Yeh, "Designing Effective Interview Chatbots: Automatic Chatbot Profiling and Design Suggestion Generation for Chatbot Debugging," Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pp. 1–12, 2021.