SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

INDORE

PROJECT REPORT

ON

"RECRUITAI: AN AI INTERVIEWER"

Submitted to "Symbiosis University of Applied Sciences, Indore"
As a Final Project report for the partial fulfillment of the award of degree of

BACHELOR OF TECHNOLOGY

IN

SCHOOL OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Submitted To:
Ms. Shruti Jain
Asst. Professor

Submitted By:
Tejas Bhati
(2020BTCS042)

SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

INDORE

CERTIFICATE

This is to certify that the project report entitled **"*RecruitAI: An AI Interviewer*"**, submitted by ***Tejas Bhati (2020BTCS042)*** student of the final year towards partial fulfillment of the degree of Bachelor of Technology in School of Computer Science and Information Technology in the year 2023-2024 Symbiosis University of Applied Sciences, Indore (M.P.)

Place:

Date:

INTERNAL EXAMINER EXTERNAL EXAMINER

SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

INDORE

RECOMMENDATION

The work entitled **"*RecruitAI: An AI Interviewer*"**, submitted by ***Tejas C. Bhati (2020BTCS042)*** student of final year Computer Science and Information Technology, towards the partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Information Technology of Symbiosis University of Applied Sciences Indore(M.P.) is a satisfactory account of their Final Project and is recommended for the award of the degree.

Endorsed By:

Dr. Durgesh Mishra

Director, SCSIT

Student Undertaking

I hereby undertake that the project work entitled **"*RecruitAI: An AI Interviewer*"** has been carried out by me from the period Jan – June 2024 and the report so prepared is a record of work done by me during my Final Project. I further declare that I have completed the Final Project in accordance with the Final Project policy of the University. This Project report is submitted towards fulfillment of my academic requirement and not for any other purpose.

I hereby undertake that the material of this Project is my original work and I have not copied anything from anywhere else. The material obtained from other sources has been duly acknowledged. I understand that if at any stage, it is found that I have indulged in any malpractice or the project and the project report has been copied or not completed by me, the university shall cancel my degree/withhold my result and appropriate disciplinary action shall be initiated against me.

Student Name and Signature Mentor Name & Signature Enrollment Number:

Date:

Director CSIT

SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

INDORE

ACKNOWLEDGEMENT

The successful completion of any work is generally not an individual effort. It is an outcome of dedicated and cumulative efforts of a number of persons, each having its own importance to the objective. This section is a value of thanks and gratitude towards all those persons who have implicitly or explicitly contributed in their own unique way towards the completion of the project. For their invaluable comments and suggestions, I wish to thank them all.

Positive inspiration and right guidance are must in every aspect of life. Especially, when we arrive at academic stage for instance. For the success of our project a number of obligations have been taken. We have performed solemn duty of expressing a heartfelt thanks to all who have endowed us with their precious perpetual guidance, suggestions and information. Any kind of help directly or indirectly has proved importance to us.

ABSTRACT

The document titled "RecruitAI: An AI Interviewer" is a final year project report submitted to the Symbiosis University of Applied Sciences, Indore. The project aims to revolutionize the traditional interview process by leveraging artificial intelligence and machine learning to autonomously conduct interviews, evaluate candidates' responses, and provide objective scores. The project's objectives include automating interview processes, ensuring objective evaluation, enabling multimodal interaction, and assessing technical competencies.

The project integrates modern technologies such as Python, Django, OpenAI API, Elevenlabs API, and Postman to create a seamless and effective interview platform. The report also includes a comprehensive analysis of the system's requirements, design, and testing procedures. It identifies potential problems and issues in the current system, such as speech recognition accuracy, code execution security, and scalability limitations, and suggests future extensions to enhance the system's performance, usability, and overall user satisfaction.

The project addresses challenges like resource intensity, subjectivity, and scalability, offering a transformative solution for organizations to improve their recruitment processes. The report concludes with an acknowledgment of the project's limitations and ethical considerations, emphasizing the need for continuous improvement and user feedback to ensure the system's long-term success and scalability.

Table of Contents

CERTIFICATE ..... **II**

RECOMMENDATION ..... **III**

ACKNOWLEDGEMENT ..... **V**

ABSTRACT ..... **VI**

Chapter 1: INTRODUCTION **1**

1.1 Introduction **2**

1.2 Literature Review **2**

1.3 Problem Definition **4**

Chapter 2: THE PROJECT **6**

2.1 Project Definition **7**

2.1.1 Objectives 7

2.1.2 Project Scope 8

Chapter 3: REQUIREMENT ANALYSIS **10**

3.1 Functional Requirements **11**

3.2 Non-Functional Requirements **12**

3.3 Use Case Specification **15**

3.3.1 Find & Document Actors 15

3.3.2 Find Use Cases 17

3.3.3 Use Case Diagrams 23

Chapter 4: DESIGN **24**

4.1 Database Design **25**

4.1.1 ER Diagram 25

4.1.2 Design Tables and Normalization 26

4.2 Class Diagram **29**

4.3 Activity Diagram **30**

4.4 Sequence Diagram **31**

Chapter 5: EXPERIMENT & TESTING **32**

5.1 Test Cases Developed **33**

5.2 Testing Used in Our Project **36**

Chapter 6: CONCLUSION **39**

6.1 Problems and Issues in Current System **40**

6.2 Future Extensions **42**

APPENDIX **44**

Appendix A: Interview Question Bank **45**

Appendix B: Sample Interview Transcripts **47**

Appendix C: Research Paper **48**

BIBLIOGRAPHY **51**

List of Figures

Figure 3.1: Student Use Case Diagram **25**

Figure 3.2: Administrator Use Case Diagram **25**

Figure 3.3: Overall System Use Case Diagram **25**

Figure 4.1: ER Diagram for RecruitAI **27**

Figure 4.2: Class Diagram for RecruitAI **31**

Figure 4.3: Activity Diagram for RecruitAI **32**

Figure 4.4: Sequence Diagram for RecruitAI **32**

List of Tables

Table: 1.1 Summary of References ***2***

Table: 3.1 Functional Requirements ***10***

Table: 3.2 Non-Functional Requirements ***11***

Table: 5.1 Test Cases for Student Module ***33***

Table: 5.2 Test Cases for Administrator Module ***34***

Table: 5.3 Testing Used in Our Project ***36***

Chapter 1:
INTRODUCTION

1.1 Introduction

RecruitAI: An AI Interviewer is a pioneering project leveraging artificial intelligence to revolutionize the traditional interviewing process. It addresses key challenges such as resource intensiveness, subjectivity, and scalability by autonomously conducting interviews, assessing candidate responses, and delivering unbiased evaluations. By reducing human effort, enhancing fairness through standardized evaluations, and increasing efficiency with prompt feedback, RecruitAI streamlines the entire hiring process. It represents a significant advancement in AI integration within human resources, offering organizations of all sizes a more efficient, fair, and scalable approach to talent evaluation and hiring decisions.

1.2 Literature Review

Table: 1.1 Summary of References

Reference Number	**Technology Used**	**Method**	**Model Utilized**	**Algorithm**	**Performance Gaps/Key Findings**
[19]	Scikit-learn	Text Processing	Supervised Learning Models	SVM	Requires continuous updates for accuracy
[20]	TensorFlow, Keras	Model Training	Neural Networks	RNN, LSTM	High computational requirements
[22]	Azure Functions	Deploy and Manage Serverless Functions	N/A	N/A	Integration challenges between backend and frontend
[23]	NLTK	Text Processing	NLU Models	Cosine Similarity	Bias in text analysis based on training data
[21]	AWS Lambda	Manage Serverless Functions	N/A	N/A	Integration of frontend and backend

[14]	Eleven Labs Speech-to-Text	Speech Recognition	N/A	N/A	Similar to [10], focus on spoken to text conversion
[4]	NLTK	Text Processing	NLU Models	Cosine Similarity	Similar to [14], focus on text analysis
[16]	Angular	Web Development	N/A	N/A	Provides frontend development resources
[17]	Python	Web Development	N/A	N/A	Provides backend development resources
[6]	Google Cloud APIs	API Integration	N/A	N/A	Access to various Google Cloud APIs
[24]	OpenAI GPT-3	Language Model	GPT-3	N/A	Advanced language processing, high resource consumption
[15]	NLTK	Source Code Access	N/A	N/A	Provides tools for NLP, important for text processing
[17]	Django	Web Development	N/A	N/A	Provides frontend development resources

1.3 Problem Definition

In today's competitive job market, organizations face significant challenges in efficiently and effectively identifying top talent. Traditional interview processes are often time consuming, resource intensive, and prone to subjectivity, leading to inconsistent evaluation outcomes. Moreover, the scalability of interview processes can become a bottleneck for organizations, especially during periods of high recruitment demand.

Challenges:

1. **Resource Intensive:** Conventional interview processes require significant human resources, including interviewers' time and effort, to conduct assessments, leading to high operational costs.
2. **Subjectivity and Bias:** Human interviewers may unintentionally introduce bias based on personal preferences, leading to inconsistent evaluation criteria and unfair treatment of candidates.
3. **Scalability:** Managing a large volume of candidates during recruitment drives or campus placements can overwhelm organizations, hindering their ability to conduct thorough assessments for each candidate.
4. **Technical Skill Evaluation:** Assessing candidates' technical competencies, especially in fields like programming, can be challenging, requiring specialized expertise and infrastructure.

Objective:

The primary objective of developing "RecruitAI: An AI Interviewer" is to address these challenges by creating an autonomous interview system powered by artificial intelligence. The goal is to streamline the interview process, improve evaluation consistency and fairness, and enhance scalability.

Key Features:

1. **Autonomous Interviewing:** Develop an AI system capable of conducting interviews autonomously, reducing the need for human intervention and minimizing operational costs.
2. **Objective Evaluation:** Implement a robust scoring mechanism based on AI algorithms to evaluate candidates' responses objectively, ensuring consistency and fairness in assessment.
3. **Multimodal Interaction:** Enable the system to interact with candidates using both spoken and written communication, providing flexibility in how candidates can respond to interview questions.
4. **Technical Competency Assessment:** Incorporate coding questions and evaluation mechanisms to assess candidates' technical skills accurately and efficiently.

Scope:

The scope of the project includes designing and implementing the AI interviewer system, integrating it with modern technologies such as Python with OpenAI API, Elevenlabs API, PostMan, and Django, and ensuring its usability and effectiveness in real-world interview scenarios.

Limitations:

1. **Technology Limitations:** The project must operate within the constraints of available technologies and APIs, such as speech recognition accuracy and computational resources for running code evaluations.
2. **Ethical Considerations:** The AI interviewer must adhere to ethical guidelines and ensure fairness, transparency, and privacy in candidate assessments.

Chapter 2:
THE PROJECT

2.1 Project Definition

2.1.1 Objectives

1. **Automate Interview Processes:**
    - **Description**: Develop an AI-driven system capable of autonomously conducting interviews without the need for human intervention.
    - **Rationale**: Automating the interview process reduces the burden on human resources, streamlines operations, and enables scalability, allowing organizations to efficiently handle large volumes of candidates.
2. **Ensure Objective Evaluation:**
    - **Description**: Implement a robust scoring mechanism based on AI algorithms to evaluate candidates' responses objectively.
    - **Rationale**: Objective evaluation reduces bias, ensures consistency in assessment criteria, and provides fair treatment to all candidates, improving the overall integrity of the recruitment process.
3. **Enable Multimodal Interaction:**
    - **Description**: Enable the system to interact with candidates using both spoken and written communication methods.
    - **Rationale**: Supporting multimodal interaction enhances the user experience, accommodates diverse communication preferences, and increases accessibility for candidates with varying needs.
4. **Assess Technical Competencies:**
    - **Description**: Incorporate coding questions and evaluation mechanisms to accurately assess candidates' technical skills.
    - **Rationale**: Assessing technical competencies is essential for roles requiring specialized skills, such as programming or software development, ensuring that candidates possess the necessary capabilities for the job.
5. **Integrate with Modern Technologies:**
    - **Description**: Utilize modern technologies such as Python, Angular, and AWS Lambda for the development and integration of the AI interviewer system.

- **Rationale**: Leveraging modern technologies enhances the functionality, performance, and scalability of the system, ensuring compatibility with existing infrastructures and facilitating seamless integration with other platforms.

6. **Ensure Usability and Effectiveness:**
   - **Description**: Design the AI interviewer system to be user friendly, intuitive, and effective in real world interview scenarios.
   - **Rationale**: Usability and effectiveness are crucial for user adoption and satisfaction, ensuring that the system meets the needs of both candidates and interviewers while delivering accurate and actionable insights.

2.1.2 Project Scope

1. **Development of AI Interviewer System:**
   - **Description**: Design and implement the core functionality of the AI interviewer system, including question generation, candidate interaction, response evaluation, and scoring mechanisms.
   - **Deliverables**: Fully functional AI interviewer system capable of autonomously conducting interviews, evaluating candidate responses, and providing objective scores.
2. **Integration with Modern Technologies:**
   - **Description**: Integrate the AI interviewer system with modern technologies such as Python, Angular, and Azure Functions for backend processing and frontend presentation.
   - **Deliverables**: Seamless integration of backend AI algorithms with Python, frontend user interface development using Angular, and deployment using Azure Functions.
3. **Support for Multimodal Interaction:**
   - **Description**: Implement support for both spoken and written interaction methods, allowing candidates to respond to interview questions using speech or text.
   - **Deliverables**: Integration of Google's Speech to Text API for speech recognition, along with text input capabilities for candidates to provide written responses.

4. **Technical Competency Assessment:**
    - **Description**: Develop coding questions and evaluation mechanisms to assess candidates' technical skills in areas such as programming and software development.
    - **Deliverables**: Implementation of coding question templates, code execution environments, and automated evaluation algorithms to assess candidates' coding proficiency.
5. **Usability Testing and Optimization:**
    - **Description**: Conduct usability testing to evaluate the user experience of the AI interviewer system and identify areas for improvement.
    - **Deliverables**: Usability testing reports, feedback analysis, and iterative refinements to enhance the system's usability, accessibility, and effectiveness.
6. **Documentation and Training Materials:**
    - **Description**: Prepare comprehensive documentation, user manuals, and training materials to guide users in deploying, using, and maintaining the AI interviewer system.
    - **Deliverables**: Documentation covering system architecture, installation instructions, usage guidelines, and troubleshooting procedures, along with training materials for interviewers and administrators.
7. **Ethical Considerations and Privacy Compliance:**
    - **Description**: Address ethical considerations and ensure compliance with privacy regulations in the design and deployment of the AI interviewer system.
    - **Deliverables**: Ethical guidelines documentation, privacy impact assessment, and implementation of privacy enhancing measures to protect candidate data and ensure fairness in evaluation.

Chapter 3: REQUIREMENT ANALYSIS

3.1 Functional Requirements

Table: 3.1 Functional Requirements

Sr. No.	Title	Description
1	Interview Process Management	• Generation and presentation of interview questions. • Management of interview session flow and timing. • Logging and tracking of candidate responses and evaluation results
2	Question Generation and Selection	• Random selection of questions from predefined question pools. • Dynamic question generation based on candidate profiles or job roles. • Support for various question types, including behavioural, technical, and situational questions.
3	Candidate Interaction	• Integration with Google's Speech-to-Text API for speech recognition. • Text input interface for candidates to provide written responses. • Natural language understanding capabilities to interpret and process candidate responses.
4	Response Evaluation	• Analysis of response content for relevance, coherence, and correctness. • Calculation of similarity scores between candidate answers and correct responses. • Coding question evaluation, including code execution and testing.
5	Scoring and Feedback	• Calculation and presentation of scores for each interview question. • Generation of detailed feedback reports highlighting strengths and areas for improvement.

		• Provision of overall performance scores and recommendations for further action.
6	Integration with Modern Technologies	• Utilization of Python with OpenAI API, Elevenlabs API, and PostMan for backend AI algorithms and processing. • Implementation of Django for frontend user interface development due to better results and compatibility. • Deployment using Azure Function for scalability and efficiency.
7	Usability and Accessibility	• Intuitive user interface design with clear navigation and feedback mechanisms • Support for accessibility features such as screen readers and keyboard navigation. • Usability testing to enhance user experience.

3.2 Non-Functional Requirements

Table: 3.2 Functional Requirements

Sr No.	**Title**	**Description**
1	Performance	• **Response Time:** The system should process and respond to user inputs (both spoken and written) within 2 seconds to ensure a smooth user experience. • **Throughput:** The system should be able to handle up to 100 simultaneous users without performance degradation. • **Scalability:** The system architecture should support scalability to accommodate increasing numbers of users and data without significant performance loss.
2	Reliability	• **Availability:** The system should have an uptime of 99.9%, ensuring it is available to users at all times.

		• **Fault Tolerance:** The system should be able to handle and recover from hardware or software failures with minimal disruption to users. • **Data Integrity:** The system must ensure that all user data, especially interview results and personal information, are accurately processed and stored without corruption.
3	Security	• **Authentication and Authorization:** Implement robust authentication mechanisms to ensure that only authorized users can access the system. • Use role-based access control (RBAC) for different user roles (e.g., students, administrators). • **Data Encryption:** All sensitive data, including user credentials and interview results, should be encrypted both in transit and at rest. • **Vulnerability Management:** Regularly update and patch the system to protect against security vulnerabilities and threats.
4	Usability	• **User Interface:** The user interface should be intuitive and user-friendly, allowing users to easily navigate and interact with the system. • **Accessibility:** Ensure that the system is accessible to users with disabilities by following WCAG (Web Content Accessibility Guidelines). • **Support and Documentation:** Provide comprehensive user manuals, FAQs, and support channels to assist users in using the system effectively.
5	Maintainability	• **Modularity:** The system should be designed with a modular architecture, making it easier to update and maintain individual components without affecting the whole system.

		• **Documentation:** Maintain detailed and up-to-date documentation for both the codebase and system architecture to facilitate maintenance and future development. • **Code Quality:** Adhere to best coding practices and conduct regular code reviews to ensure high code quality and maintainability.
6	Portability	• **Platform Independence:** The system should be designed to run on multiple platforms (Windows, macOS, Linux) without requiring significant modifications. • **Browser Compatibility:** The frontend application should be compatible with all major web browsers (Chrome, Firefox, Safari, Edge) to ensure a consistent user experience.
7	Interoperability	• **API Integration:** The system should provide well-defined APIs to facilitate integration with other systems and services, such as educational platforms or third-party interview tools. • **Data Export and Import:** Enable easy export and import of data in standard formats (e.g., CSV, JSON) to allow seamless data exchange with other applications.
8	Efficiency	• **Resource Utilization:** The system should optimize the use of computational resources (CPU, memory, storage) to provide cost-effective performance. • **Power Consumption:** For cloud deployment, the system should be designed to minimize power consumption to reduce operational costs and environmental impact.
9	Compliance	• **Legal and Regulatory Compliance:** Ensure that the system complies with all relevant legal and regulatory requirements, such as data protection laws (e.g., GDPR, CCPA). • **Industry Standards:** Adhere to industry standards and best practices for software development, security, and data management.

3.3 Use Case Specification

3.3.1 Find & Document Actors

1. Primary Actors:

These are the main users who directly interact with the system to achieve their goals.

1. **Student**

   Description: A person who participates in the interview process. Students answer questions, both spoken and written and receive scores.

   Interactions:

   - Logging in to the system.
   - Taking the interview by answering questions.
   - Receiving and reviewing scores.
   - Getting feedback on performance.

2. **Administrator**

   Description: A person responsible for managing the system, including user accounts and interview settings.

   Interactions:

   - Creating and managing student accounts.
   - Setting up interview questions.
   - Monitoring system performance and usage.
   - Accessing and analyzing interview results.

2. Secondary Actors:

These are entities that support the primary actors in achieving their goals. They usually interact with the system indirectly.

1. **System Database**

   Description: A database that stores user information, interview questions, answers, and scores.

   Interactions:

   - Storing and retrieving user data.
   - Maintaining interview questions and correct answers.

- Recording interview responses and scores.

2. **Speech Recognition Service**

   Description: An external service used to convert spoken answers into text.

   Interactions:

   - Receiving audio input from students.
   - Processing audio and converting it to text.
   - Sending the text back to the system for further processing.

3. **Code Execution Environment**

   Description: An environment where students' coding answers are executed and evaluated.

   Interactions:

   - Receiving and executing code written by students.
   - Evaluating code for correctness and efficiency.
   - Sending execution results and scores back to the system.

4. **AI Scoring Engine**

   Description: The component responsible for calculating scores based on the similarity of students' answers to the correct answers using tokenization and cosine similarity.

   Interactions:

   - Receiving textual answers from students.
   - Comparing answers to correct answers using tokenization and cosine similarity.
   - Calculating and sending scores back to the system.

3. External Systems

Systems that interact with RecruitAI to provide additional functionalities.

1. **Azure Functions**

   Description: A serverless compute service offered by Microsoft Azure that enables the execution of event-driven code without the need to manage infrastructure.

   Interactions:

   - Hosting and running backend functions.
   - Handling requests from the front end.
   - Providing seamless integration between frontend and backend.

- Scalability and Reliability
- Integration with Azure Services

4. Supporting Actors

These actors provide ancillary functions that support the primary actors and the system.

1. **Logging and Monitoring Services**

   Description: Services used for tracking system performance and usage, and detecting issues.

   Interactions:

   - Collecting logs from various components.
   - Monitoring system health and performance.
   - Sending alerts and notifications to administrators in case of issues.

2. **Notification Service**

   Description: A service used for sending notifications to users.

   Interactions:

   - Sending interview invitations and reminders to students.
   - Notifying students about their scores and feedback.
   - Alerting administrators about important events or issues.

3.3.2 Find Use Cases

1. Use Case: User Registration

Actors: Student, System Database

Description: This use case describes how a new student registers for an account in the system.

Preconditions:

- The student must have access to the internet.
- The system is online and accessible.

Main Flow:

1. The student navigates to the registration page.

2. The student fills out the registration form with the required information (name, email, password, etc.).
3. The system validates the input data.
4. The system stores the new user information in the System Database.
5. The system sends a confirmation email to the student.
6. The student confirms the email address by clicking the link in the email.

Postconditions:

- The student's account is created and stored in the System Database.
- The student can log in to the system using their new credentials.

Alternate Flows:

- If the email is already in use, the system prompts the student to use a different email.
- If the input data is invalid, the system prompts the student to correct the errors.

2. Use Case: User Login

Actors: Student, System Database

Description: This use case describes how a student logs into the system.

Preconditions:

- The student must have a registered account.
- The system is online and accessible.

Main Flow:

1. The student navigates to the login page.
2. The student enters their email and password.
3. The system validates the credentials against the System Database.
4. The system logs the students in and redirects them to the dashboard.

Postconditions:

- The student is logged in and can access the interview functionalities.

Alternate Flows:

- If the credentials are incorrect, the system displays an error message.
- If the account is not confirmed, the system prompts the student to confirm their email.

3. Use Case: Start Interview

Actors: Student, System Database

Description: This use case describes how a student starts an interview session.

Preconditions:

- The student must be logged in.
- There must be available interview questions in the System Database.

Main Flow:

1. The student navigates to the interview section.
2. The student clicks on the "Start Interview" button.
3. The system retrieves the first question from the System Database.
4. The system presents the question to the student.

Postconditions:

- The interview session is initiated, and the first question is displayed.

Alternate Flows:

- If no questions are available, the system displays a message indicating that there are no interviews available.

4. Use Case: Answer Interview Question

Actors: Student, System Database, Speech Recognition Service, AI Scoring Engine

Description: This use case describes how a student answers a question during the interview.

Preconditions:

- The student must be in an active interview session.

Main Flow:

1. The student provides a spoken answer.
2. The system captures the audio input and sends it to the Speech Recognition Service.
3. The Speech Recognition Service converts the audio to text and returns it to the system.
4. The system sends the text answer to the AI Scoring Engine.
5. The AI Scoring Engine calculates the similarity score and sends it back to the system.

6. The system displays the next question or the interview results if it was the last question.

Postconditions:

- The student's answer is processed and scored.

Alternate Flows:

- If the speech recognition fails, the student is prompted to type their answer manually.
- If the AI Scoring Engine fails, the system logs the error and prompts the student to retry the question.

5. Use Case: Submit Coding Solution

Actors: Student, System Database, Code Execution Environment

Description: This use case describes how a student submits a coding solution during the interview.

Preconditions:

- The student must be in an active interview session with a coding question.

Main Flow:

1. The student receives a coding question.
2. The student writes and submits their code in the provided editor.
3. The system sends the code to the Code Execution Environment.
4. The Code Execution Environment executes the code and evaluates it.
5. The Code Execution Environment returns the execution results and scores to the system.
6. The system displays the results to the student and proceeds to the next question or completes the interview.

Postconditions:

- The student's coding solution is executed and scored.

Alternate Flows:

- If the code execution environment fails, the system provides an error message and prompts the student to retry.

6. Use Case: View Interview Results

Actors: Student, System Database

Description: This use case describes how a student views their interview results after completing the interview.

Preconditions:

- The student must have completed an interview.

Main Flow:

1. The student navigates to the results section.
2. The system retrieves the interview results from the System Database.
3. The system displays the results, including scores and feedback.

Postconditions:

- The student views their scores and feedback.

Alternate Flows:

- If the results are not available, the system displays a message indicating the results are being processed.

7. Use Case: Manage Users

Actors: Administrator, System Database

Description: This use case describes how an administrator manages user accounts.

Preconditions:

- The administrator must be logged in with appropriate privileges.

Main Flow:

1. The administrator navigates to the user management section.
2. The administrator adds, edits, or deletes user accounts.
3. The system updates the user information in the System Database.

Postconditions:

- User accounts are successfully managed.

Alternate Flows:

- If the administrator tries to delete a nonexistent user, the system displays an error message.

8. Use Case: Configure Interview Questions

Actors: Administrator, System Database

Description: This use case describes how an administrator configures interview questions.

Preconditions:

- The administrator must be logged in with appropriate privileges.

Main Flow:

1. The administrator navigates to the interview configuration section.
2. The administrator adds, edits, or deletes interview questions.
3. The system updates the interview questions in the System Database.

Postconditions:

- Interview questions are successfully configured.

Alternate Flows:

- If the administrator tries to delete a question in use, the system displays a warning message.

3.3.3 Use Case Diagrams

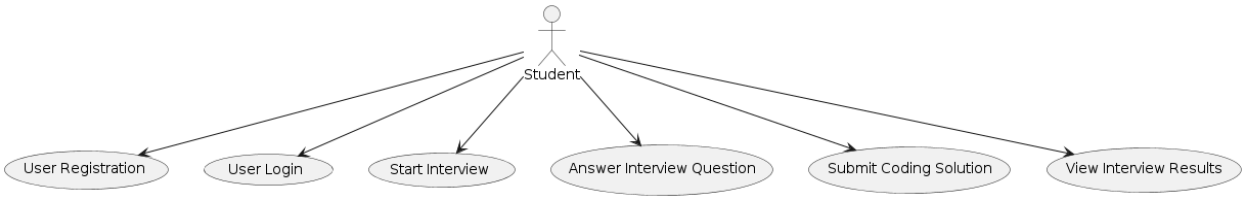

Figure 3.1: Student Use Case Diagram

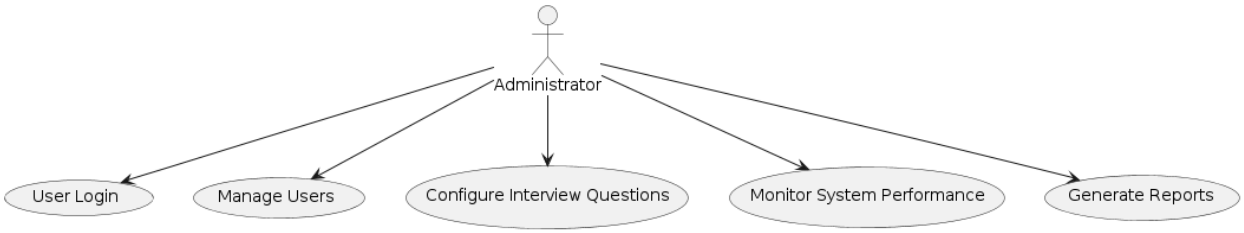

Figure 3.2: Administrator Use Case Diagram

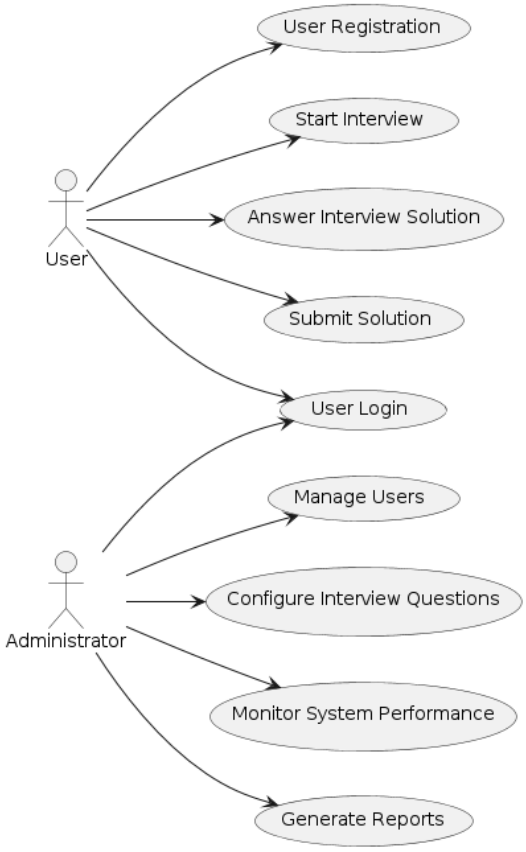

Figure 3.3: Overall System Use Case Diagram

Chapter 4: DESIGN

4.1 Database Design

4.1.1 ER Diagram

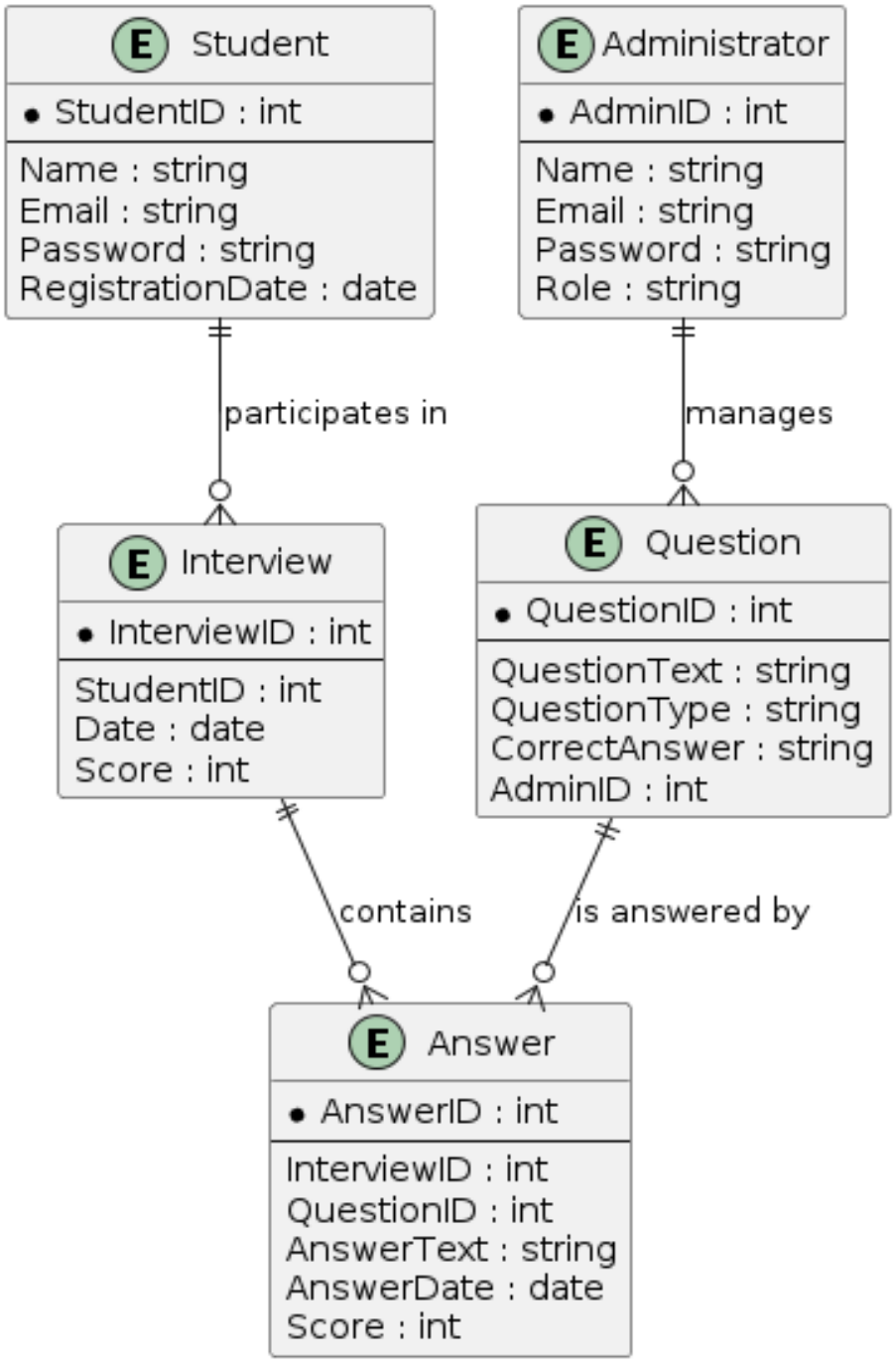

Figure 4.1: ER Diagram for RecruitAI

4.1.2 Design Tables and Normalization

Database Tables Design

1. **Student**
    - **StudentID**: Unique identifier for each student (Primary Key).
    - **Name:** Name of the student.
    - **Email:** Email address of the student, must be unique.
    - **Password:** Password for student login.
    - **RegistrationDate:** Date when the student registered.

```
CREATE TABLE Student (
   StudentID INT PRIMARY KEY AUTO_INCREMENT,
   Name VARCHAR(100) NOT NULL,
   Email VARCHAR(100) UNIQUE NOT NULL,
   Password VARCHAR(100) NOT NULL,
   RegistrationDate DATE NOT NULL
);
```

2. **Administrator**
    - **AdminID:** Unique identifier for each administrator (Primary Key).
    - **Name:** Name of the administrator.
    - **Email:** Email address of the administrator, must be unique.
    - **Password:** Password for administrator login.
    - **Role:** Role of the administrator in the system.

```
CREATE TABLE Administrator (
   AdminID INT PRIMARY KEY AUTO_INCREMENT,
   Name VARCHAR(100) NOT NULL,
   Email VARCHAR(100) UNIQUE NOT NULL,
   Password VARCHAR(100) NOT NULL,
   Role VARCHAR(50) NOT NULL
);
```

3. **Interview**
    - **InterviewID:** Unique identifier for each interview (Primary Key).
    - **StudentID:** Identifier of the student taking the interview (Foreign Key referencing StudentID).
    - **Date:** Date when the interview was taken.
    - **Score:** Score achieved by the student in the interview.

```
CREATE TABLE Interview (
  InterviewID INT PRIMARY KEY AUTO_INCREMENT,
  StudentID INT NOT NULL,
  Date DATE NOT NULL,
  Score INT,
  FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);
```

4. **Question**
    - **QuestionID:** Unique identifier for each question (Primary Key).
    - **QuestionText:** The text of the question.
    - **QuestionType:** Type of the question (e.g., spoken, written, coding).
    - **CorrectAnswer:** Correct answer for the question.
    - **AdminID:** Identifier of the administrator who created the question (Foreign Key referencing AdminID).

```
CREATE TABLE Question (
  QuestionID INT PRIMARY KEY AUTO_INCREMENT,
  QuestionText TEXT NOT NULL,
  QuestionType VARCHAR(50) NOT NULL,
  CorrectAnswer TEXT NOT NULL,
  AdminID INT NOT NULL,
  FOREIGN KEY (AdminID) REFERENCES Administrator(AdminID)
);
```

5. **Answer**
    - **AnswerID:** Unique identifier for each answer (Primary Key).
    - **InterviewID:** Identifier of the interview (Foreign Key referencing InterviewID).
    - **QuestionID:** Identifier of the question (Foreign Key referencing QuestionID).
    - **AnswerText:** The text of the answer provided by the student.
    - **AnswerDate:** Date when the answer was given.
    - **Score:** Score achieved for the answer.

```
CREATE TABLE Answer (
  AnswerID INT PRIMARY KEY AUTO_INCREMENT,
  InterviewID INT NOT NULL,
  QuestionID INT NOT NULL,
  AnswerText TEXT NOT NULL,
  AnswerDate DATE NOT NULL,
  Score INT,
  FOREIGN KEY (InterviewID) REFERENCES Interview(InterviewID),
  FOREIGN KEY (QuestionID) REFERENCES Question(QuestionID)
);
```

Normalization

The tables are designed to be in the third normal form (3NF) i.e. there are no transitive dependencies; all non key attributes are only dependent on the primary key.

4.2 Class Diagram

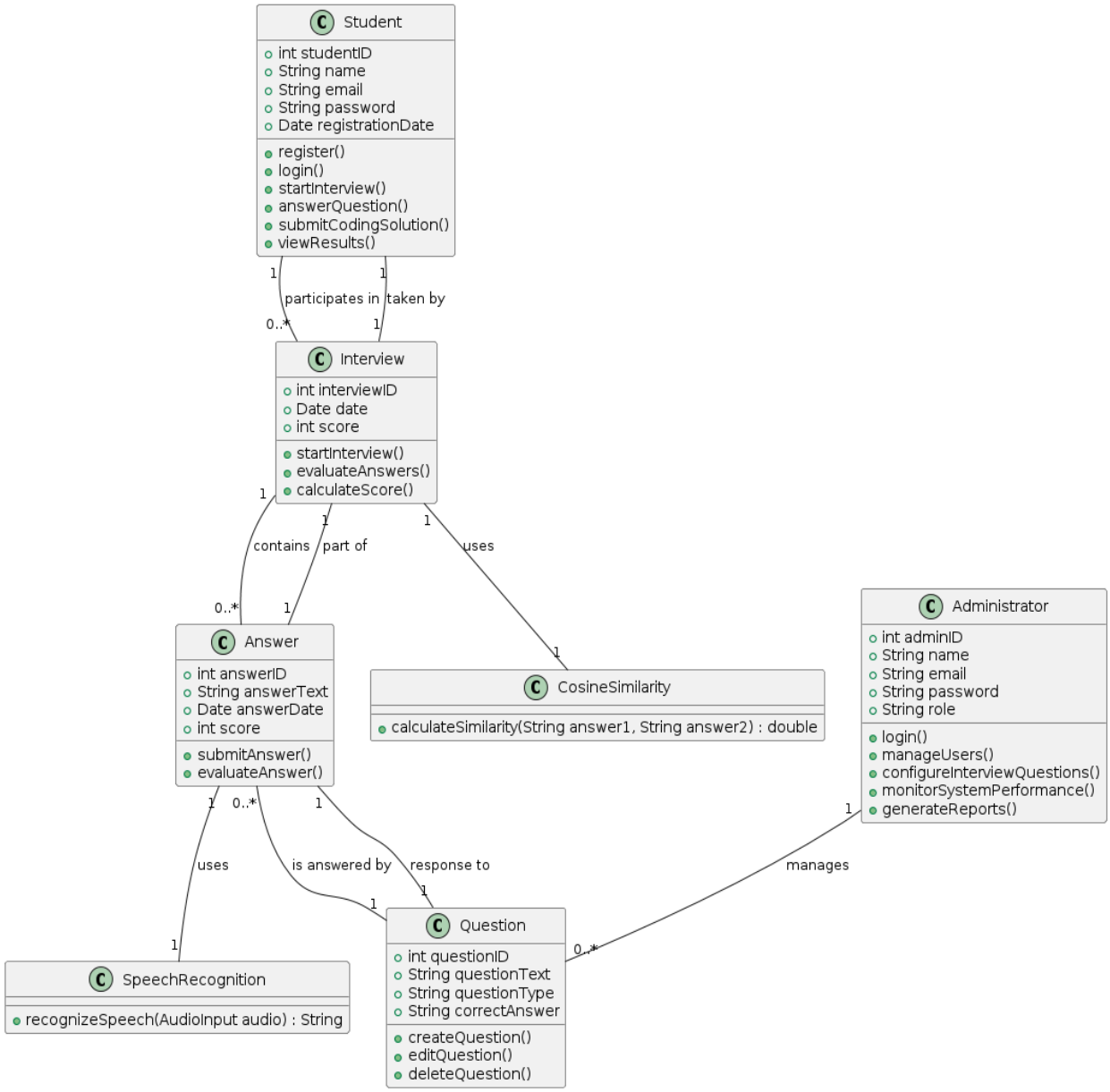

Figure 4.2: Class Diagram for RecruitAI

4.3 Activity Diagram

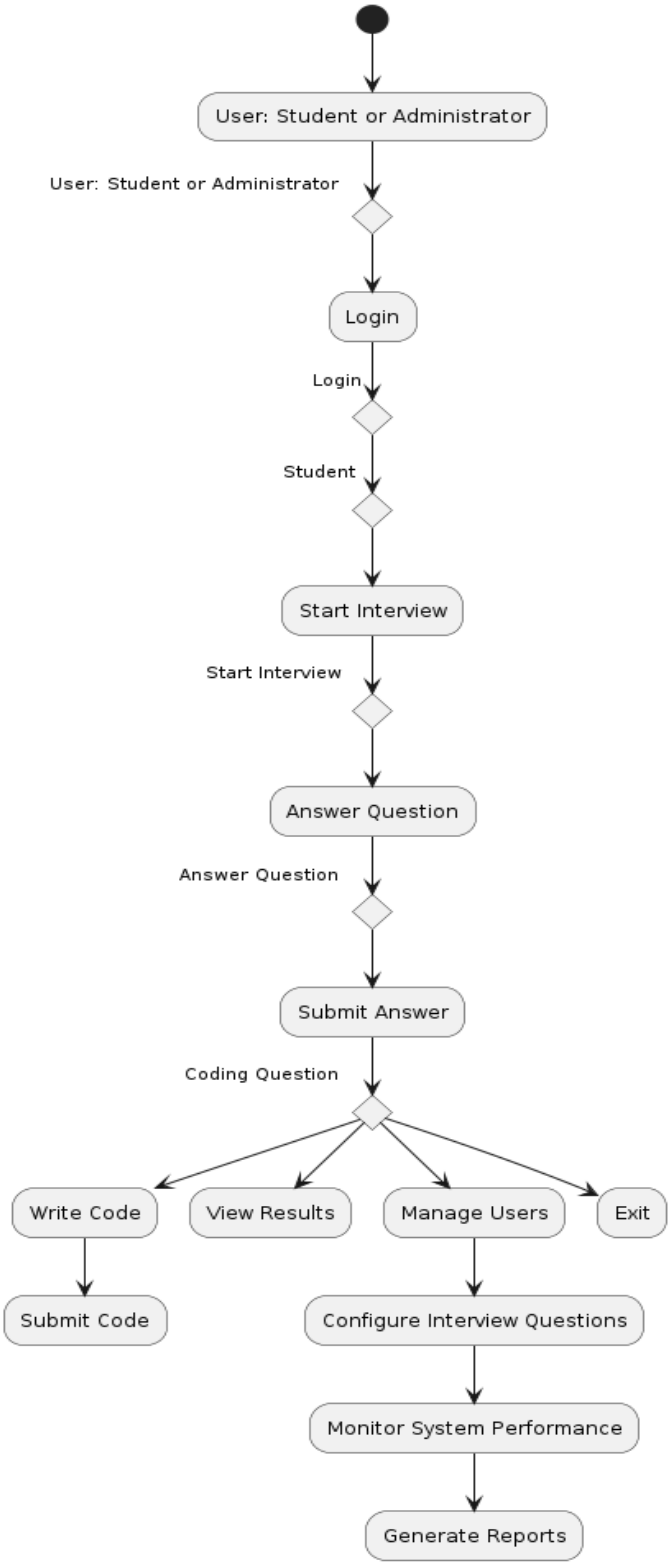

Figure 4.3: Activity Diagram for RecruitAI

4.4 Sequence Diagram

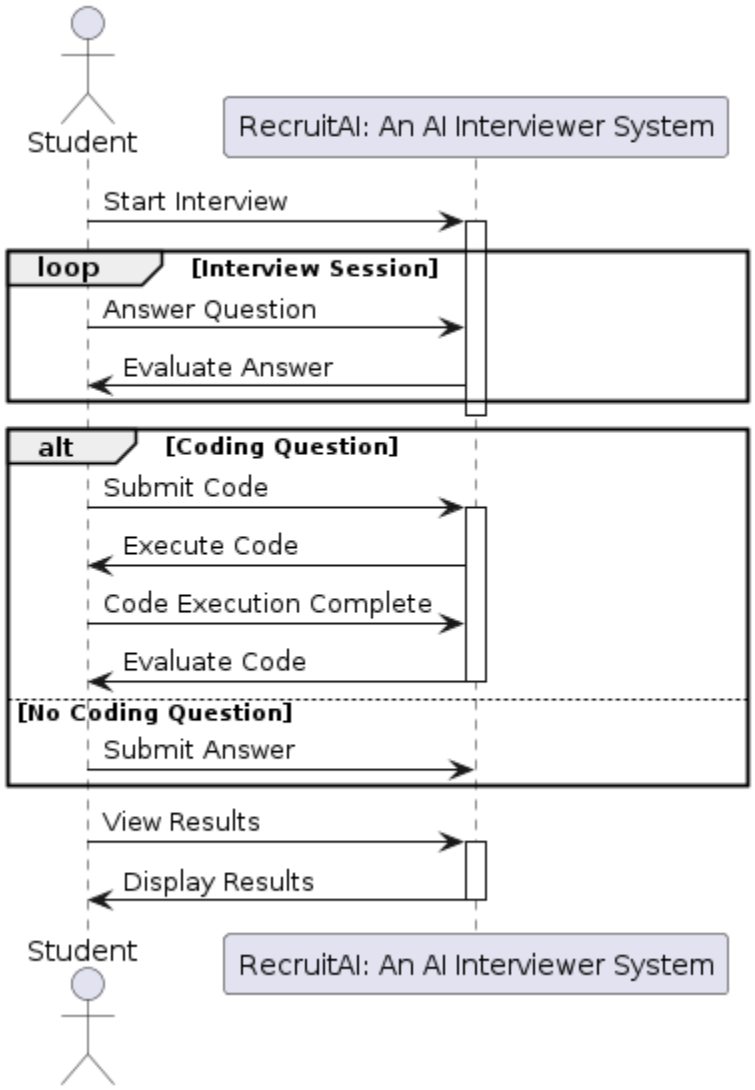

Figure 4.4: Sequence Diagram for RecruitAI

Chapter 5: EXPERIMENT & TESTING

5.1 Test Cases Developed

Table 5.1: Test Cases for Student Module

Sr No	Test Case	Objective	Preconditions	Steps	Expected Result
1	Student Registration	To verify that a student can successfully register in the system.	N/A	1. Navigate to the registration page. 2. Enter valid student details (name, email, password). 3. Click on the registration button.	The student should be registered successfully and redirected to the login page.
2	Student Login	To verify that a registered student can log in to the system.	N/A	1. Navigate to the login page. 2. Enter valid student email and password. 3. Click on the login button.	The student should be logged in successfully and redirected to the dashboard.
3	Start Interview	To verify that a student can start an interview session.	Student is logged in	1. Navigate to the interview section. 2. Click on the "Start Interview" button.	An interview session should be initiated successfully.
4	Answer Questions	To verify that a student can answer	Interview session is ongoing	1. Read the question. 2. Provide an answer.	The student's answer should be

		interview questions.			recorded successfully.
5	View Interview Results	To verify that a student can view interview results after completing the interview.	The interview session is completed	1. Navigate to the interview results section.	The student should be able to view their interview score and feedback.

Table 5.2: Test Cases for Administrator Module

Sr No	**Test Case**	**Objective**	**Preconditions**	**Steps**	**Expected Result**
1	Administrator Login	To verify that an administrator can log in to the system.	N/A	2. Navigate to the login page. 3. Enter valid administrator email and password. 4. Click on the login button.	The administrator should be logged in successfully and redirected to the dashboard.
2	Manage Users	To verify that an administrator can manage user accounts.	Administrator is logged in	1. Navigate to the user management section. 2. Search for a user by email or name. 3. Perform actions like edit, delete, or suspend user accounts.	The administrator should be able to manage user accounts successfully.

3	Configure Interview Questions	To verify that an administrator can configure interview questions.	Administrator is logged in	1. Navigate to the question management section. 2. Add, edit, or delete interview questions.	The administrator should be able to configure interview questions successfully.
4	Monitor System Performance	To verify that an administrator can monitor system performance metrics.	Administrator is logged in	1. Navigate to the performance monitoring section. 2. View system metrics such as CPU usage, memory usage, and response times.	The administrator should be able to monitor system performance effectively.
5	Generate Reports	To verify that an administrator can generate reports.	Administrator is logged in	1. Navigate to the reporting section. 2. Select the type of report to generate (e.g., interview performance, user activity). 3. Specify report parameters (e.g., date range, user groups). 4. Generate the report.	The administrator should be able to generate reports with accurate data.

5.2 Testing Used in Our Project

Table: 5.3 Testing Used in Our Project

Sr No	Testing Type	Objective	Tools/Frameworks	Examples	Expected Outcome
1	Unit Testing	To test individual units or components of the system in isolation.	Python's built-in unittest or pytest framework.	• Test tokenization functionality • Test cosine similarity calculation • Test speech recognition and text conversion • Test code execution and evaluation for coding questions	Each unit works correctly and independently.
2	Integration Testing	To test the interaction between different modules or components.	Python's unittest.mock library, Postman for API testing.	• Test backend Python code and Django frontend integration • Test communication between speech recognition and answer evaluation modules • Test AWS Lambda functions integration	Modules interact correctly and data flows smoothly between them.

3	System Testing	To test the behavior of the entire system as a whole.	Selenium for web app testing, PyAutoGUI for GUI automation.	• Test entire interview process from login to viewing results • Test various student interaction scenarios during an interview • Test administrator functionalities such as user management and question configuration	The entire system works correctly under various conditions.
4	Performance Testing	To evaluate performance characteristics under different load conditions.	Apache JMeter, Locust, custom scripts for load testing.	• Measure response time under normal and peak loads • Evaluate scalability with large number of concurrent users • Test system performance during multiple simultaneous interviews	System performs well under expected and peak loads.
5	User Acceptance Testing (UAT)	To validate whether the system meets requirements and is acceptable to end-users.	Manual testing by end-users, feedback collection mechanisms.	• Involve students to test and provide feedback on interview process • Engage administrators to validate and provide feedback on	The system is usable and meets the needs of end-users.

				administrative functionalities • Gather feedback for identifying issues or improvements before release	
6	Security Testing	To identify and address security vulnerabilities in the system.	Static code analysis tools like Bandit, OWASP ZAP for dynamic testing.	• Test for SQL injection vulnerabilities • Perform authentication and authorization testing • Validate data encryption mechanisms	The system is secure and protects against vulnerabilities.
7	Regression Testing	To ensure new changes or bug fixes do not adversely affect existing functionality.	Automated test suites integrated into the CI/CD pipeline.	• Rerun existing test cases after new features or changes • Verify bug fixes do not introduce new issues Ensure changes in one module do not affect other modules	Existing functionality remains unaffected by new changes.

Chapter 6:
CONCLUSION

6.1 Problems and Issues in Current System

Identifying problems and issues in the current system is crucial for improving its performance, usability, and overall user satisfaction. Here are some potential problems and issues that may exist in the "RecruitAI: An AI Interviewer" system:

1. **Speech Recognition Accuracy:**

   Problem: The speech recognition component may have low accuracy, leading to incorrect transcription of spoken answers.

   Issue: This can result in incorrect evaluation of student responses and affect the overall assessment accuracy.

2. **Code Execution Security:**

   Problem: The system allows students to submit and execute code for coding questions.

   Issue: Without proper security measures, this could potentially lead to security vulnerabilities such as code injection attacks or unauthorized access to system resources.

3. **Scalability Limitations:**

   Problem: The system may not be designed to handle a large number of concurrent users or interviews.

   Issue: This could lead to performance degradation, slow response times, or system crashes during peak usage periods.

4. **Lack of Feedback Mechanisms:**

   Problem: The system may not provide adequate feedback to students on their interview performance.

   Issue: Without feedback, students may not know how to improve, leading to frustration and disengagement from the platform.

5. **Administrator Complexity:**

   Problem: The administrative functionalities of the system may be complex and difficult to use.

   Issue: This could result in errors or delays in managing users, configuring questions, or generating reports, impacting the efficiency of administrators.

6. **Inadequate User Documentation:**

   Problem: The system may lack comprehensive user documentation or tutorials.

   Issue: Users, especially new students or administrators, may struggle to understand how to use the system effectively, leading to confusion and frustration.

7. **Performance Bottlenecks:**

   Problem: Certain components of the system, such as database queries or algorithm computations, may be inefficient.

   Issue: This could cause delays in processing user requests, resulting in poor user experience and decreased system responsiveness.

8. **Compatibility Issues:**

   Problem: The system may not be fully compatible with all web browsers or devices.

   Issue: Users may encounter layout or functionality issues when accessing the system from different browsers or devices, leading to inconsistency in user experience.

9. **Data Privacy Concerns:**

   Problem: The system may collect and store sensitive user data without adequate privacy protection measures.

   Issue: This could violate privacy regulations or expose user data to unauthorized access, leading to legal and reputational risks.

10. **Lack of Regular Updates:**

    Problem: The system may not receive regular updates or maintenance.

    Issue: This could result in outdated features, security vulnerabilities, or compatibility issues with newer technologies, diminishing the system's reliability and relevance over time.

6.2 Future Extensions

Planning for future extensions is crucial for the long-term success and scalability of the "RecruitAI: An AI Interviewer" system. Here are some potential areas for future extensions and enhancements:

1. **Enhanced Speech Recognition:** Integrate advanced speech recognition technologies, such as natural language understanding (NLU) and sentiment analysis, to improve accuracy and interpret spoken answers more effectively.
2. **Machine Learning for Answer Evaluation:** Implement machine learning algorithms to analyze student responses and provide more intelligent feedback and scoring based on contextual understanding and similarity to model answers.
3. **Real-time Collaboration:** Enable real-time collaboration features, allowing multiple interviewers to conduct interviews simultaneously and collaborate on evaluating student responses.
4. **Personalized Learning Paths:** Develop algorithms to analyze student performance data and recommend personalized learning paths or resources tailored to each student's strengths, weaknesses, and learning goals.
5. **Code Assessment Enhancements:** Enhance the code assessment capabilities by incorporating static code analysis tools, unit testing frameworks, and code review functionalities to provide comprehensive feedback on coding assignments.
6. **Virtual Reality Interviews:** Explore the integration of virtual reality (VR) technology to create immersive interview environments, allowing students to practice interview skills in realistic scenarios and receive feedback in a simulated setting.
7. **Gamification and Engagement Features:** Incorporate gamification elements such as badges, leaderboards, and achievement rewards to increase student engagement and motivation in using the platform.
8. **Adaptive Questioning:** Implement adaptive questioning algorithms that dynamically adjust the difficulty level and type of questions based on the student's performance and learning progress.
9. **Multi-language Support:** Extend language support beyond English to accommodate students and administrators from diverse linguistic backgrounds, enabling a broader user base to benefit from the system.

10. **Mobile Application:** Develop a mobile application version of the system to provide on-the-go access for students and administrators, facilitating convenient interview preparation and management from mobile devices.

11. **Analytics and Insights:** Enhance reporting and analytics capabilities to provide administrators with detailed insights into student performance trends, interview effectiveness, and system usage patterns, enabling data-driven decision-making and continuous improvement.
12. **Integration with Learning Management Systems (LMS):** Integrate with existing learning management systems to streamline user authentication, data synchronization, and course management processes, providing a seamless experience for users already using LMS platforms.
13. **Accessibility Features:** Incorporate accessibility features such as screen reader compatibility, keyboard navigation, and alternative text descriptions to ensure the platform is accessible to users with disabilities and conforms to accessibility standards.
14. **Social Integration:** Integrate social media sharing and collaboration features to allow students to share their interview experiences, achievements, and feedback with peers and mentors, fostering a supportive learning community.
15. **Continuous Improvement and Feedback Mechanisms:** Implement mechanisms for collecting user feedback and suggestions, establishing a feedback loop to continuously improve system features, usability, and user experience based on user input and evolving needs.

APPENDIX

Appendix A: Interview Question Bank

Technical Questions:

1. Explain the difference between HTML and CSS.
2. What is the difference between JavaScript and Python?
3. Describe the concept of object-oriented programming (OOP).
4. What are the advantages of using version control systems like Git?
5. Explain the difference between SQL and NoSQL databases.
6. What is the purpose of the viewport meta tag in web development?
7. How does a CDN (Content Delivery Network) improve website performance?
8. What are the different types of HTTP requests and their purposes?
9. Describe the role of a web server in client-server architecture.
10. Explain the concept of responsive design and its importance in modern web development.

Behavioral Questions:

1. Describe a situation where you had to work as part of a team to achieve a common goal.
2. Can you provide an example of a time when you faced a challenging situation and how you resolved it?
3. Describe a successful project you completed and the role you played in it.
4. How do you handle constructive criticism from peers or supervisors?
5. Can you discuss a time when you had to prioritize multiple tasks with tight deadlines?
6. What motivates you to succeed in your academic or professional endeavors?
7. How do you handle stress and pressure in a fast-paced work environment?
8. Describe a time when you had to adapt to a significant change in your work or personal life.
9. How do you approach conflict resolution when working with colleagues or team members?
10. Can you provide an example of a time when you demonstrated leadership qualities?

Situational Questions:

1. Imagine you're leading a team project, and a team member isn't pulling their weight. How would you handle this situation?
2. You're working on a project, and the requirements suddenly change. How do you adapt to these changes?
3. You discover a critical bug in your code right before a deadline. How do you handle this situation?
4. You're assigned to work with a difficult colleague. How do you approach collaboration with them?
5. Describe a situation where you had to deal with a dissatisfied client or customer. How did you handle it?
6. What steps would you take to resolve a technical issue you encounter during a project?
7. How do you ensure effective communication and collaboration within a distributed team?
8. You're tasked with implementing a new feature with a tight deadline. How do you prioritize your tasks and manage your time effectively?
9. Describe a situation where you had to make a difficult decision that had an impact on your team or organization.
10. How do you stay updated on the latest trends and advancements in your field of expertise?

Coding Questions:

1. Write a Python function to calculate the factorial of a number.
2. Implement a bubble sort algorithm in Java.
3. Write SQL queries to retrieve data from a database table.
4. Create a simple HTML form with input fields for name, email, and message.
5. Implement a function in JavaScript to reverse a string.
6. Write a Python program to find the sum of all even numbers in a list.
7. Implement a binary search algorithm in any programming language of your choice.
8. Write a SQL query to find the second highest salary from an Employee table.
9. Create a CSS layout for a responsive navigation bar.
10. Implement a recursive function to compute the Fibonacci sequence in Python.

Appendix B: Sample Interview Transcripts

Interview 1: Technical Interview

Question: Explain the difference between HTML and CSS.

Response: HTML is used for structuring web content, while CSS is used for styling and formatting that content.

Interview 2: Behavioral Interview

Question: Can you provide an example of a time when you faced a challenging situation and how you resolved it?

Response: Sure, in my previous job, we encountered a major technical issue right before a product launch. I collaborated with the development team to troubleshoot the problem and implemented a workaround solution, which allowed us to meet the deadline successfully.

Interview 3: Situational Interview

Question: Imagine you're leading a team project, and a team member isn't pulling their weight. How would you handle this situation?

Response: I would approach the team member privately to discuss their workload and any challenges they may be facing. If necessary, I would offer assistance or resources to help them meet their responsibilities. If the issue persists, I will escalate the matter to higher management for further action.

Interview 4: Coding Interview

Question: Write a Python function to calculate the factorial of a number.

Response:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n  factorial(n-1)
```

Interview 5: Domain-specific Interview

Question: In web development, what is the purpose of the viewport meta tag?

Response: The viewport meta tag is used to control the layout and scaling of a web page on different devices and screen sizes.

BIBLIOGRAPHY

Natural Language Processing (NLP):

[1] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., Pearson, 2021.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *ArXiv preprint arXiv:1301.3781*, 2013.

[3] D. Jurafsky et al., "Speech and Language Processing," Prentice Hall, 2008.

[4] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of NAACL-HLT 2019*, 2019.

[6] H. Taylor, B. Kang, and K. Swarnkar, "A Review of Recent Advancements in Natural Language Processing," *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Speech Recognition:

[7] D.Amodei et al., "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Machine Learning and Deep Learning:

[8] Vaswani et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, 2017.

[9] Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[10] R. Wang et al., "Advancements in Transformer Architectures for NLP," *ArXiv preprint arXiv:2204.02152*, 2022.

Ethical Considerations and Bias Mitigation:

[11] R. Binns, "Fairness in Machine Learning: Lessons from Political Philosophy," in *Proceedings of the 2018 Conference on Fairness, Accountability, and Transparency*, 2018.

Books:

[12] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed., Pearson, 2021.

[13] P. Flach, *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*, Cambridge University Press, 2012.

Websites and Online Resources:

[14] Eleven Labs API Documentation. Available: https://elevenlabs.io/docs . Accessed: March 3, 2024.

[15] OpenAI API Documentation. Available: https://beta.openai.com/docs. Accessed: May 1, 2024.

[16] Angular Documentation. [Online]. Available: https://angular.io/docs. [Accessed: March 22, 2024].

[17] Python Software Foundation, "Python Documentation," [Online]. Available: https://docs.python.org/3/. [Accessed: March 5, 2024].

[18] Django Documentation. Available: https://docs.djangoproject.com/en/stable/. Accessed: March 2, 2024.

[19] Scikit-learn Documentation. Available: https://scikit-learn.org/stable/documentation.html. Accessed: April 22, 2024.

[20] TensorFlow and Keras Documentation. Available: https://www.tensorflow.org/guide. Accessed: April 25, 2024.

[21] AWS Lambda Documentation. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. Accessed: April 26, 2024.

[22] Microsoft Azure, "Azure Functions Documentation," 2024. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-functions/.

[23] Natural Language Toolkit (NLTK) Documentation. Available: https://www.nltk.org/. Accessed: April 5, 2024.

[24] Google Cloud Speech-to-Text Documentation. [Online]. Available: https://cloud.google.com/speech-to-text/docs . [Accessed: May 2, 2024].