

MockPro: AI Mock Interviewer

Chapter 1 - Introduction

1.1 Background

In the rapidly evolving job market, the ability to excel in interviews has become increasingly crucial for job seekers. Traditional interview processes often rely on human interviewers, who can be limited in their availability, consistency, and objectivity. This project aims to address these challenges by introducing an AI-powered mock interview system that provides a comprehensive, personalized, and scalable solution for candidates to prepare for their dream jobs.

The background of this project is rooted in the growing demand for innovative tools that can help job seekers enhance their interview skills and confidence. As the job market becomes more competitive, candidates are constantly seeking new ways to stand out and showcase their abilities effectively. However, the traditional interview format can be daunting, with limited opportunities to practice and receive personalized feedback.

This project was inspired by the recognition that advancements in artificial intelligence and natural language processing can be leveraged to create an intelligent, automated interview system. By harnessing the power of AI, the project aims to provide job seekers with a platform that can simulate real-world interview scenarios, assess their responses, and offer valuable insights to help them improve their performance.

The concept of AI-powered interview assistants is not entirely new, with some existing solutions focusing on providing general interview tips or practicing common questions. However, this project seeks to go beyond these basic functionalities by creating a comprehensive, role-specific interview experience that closely mimics the real-world interview process.

One of the key drivers behind this project is the need to democratize access to high-quality interview preparation resources. Traditionally, job seekers have had to rely on limited opportunities to practice with friends, family, or career coaches, which can be time-consuming and inconsistent. By leveraging AI technology, this project aims to provide a scalable and accessible solution that can be used by job seekers from diverse backgrounds, regardless of their geographical location or financial resources.

The project's background is also influenced by the growing emphasis on soft skills and behavioral-based interviews in the modern job market. Many employers are increasingly focused on assessing a candidate's problem-solving abilities, communication skills, and cultural fit, in addition to their technical expertise. This project aims to address this shift by designing an interview system that can evaluate a wide range of skills, including both technical and non-technical competencies.

Furthermore, the project's background is shaped by the recognition that traditional interview formats can be prone to biases and inconsistencies. By leveraging AI-powered assessment, this project aims to create a more objective and standardized evaluation process, helping to level the playing field for job seekers and promote greater diversity and inclusion in the hiring process.

Overall, the background of this project is a response to the evolving needs of the job market, the desire to empower job seekers with effective interview preparation tools, and the potential of AI technology to transform the traditional interview process. By creating a comprehensive, personalized, and scalable AI-powered mock interview system, this project seeks to enhance the interview preparedness and success of job seekers, ultimately contributing to their career advancement and the growth of the workforce.

1.2 Objectives

The primary objectives of this project are to create an AI-powered mock interview system that provides a comprehensive, personalized, and scalable solution for job seekers to prepare for their dream jobs. The key objectives of this project are as follows:

1. **Simulate Real-World Interview Scenarios:** The project aims to create a mock interview environment that closely replicates the experience of a real-world job interview, including the format, questions, and interactions. This objective ensures that the training experience is as realistic and relevant as possible for the users.
2. **Personalize the Interview Experience:** By leveraging AI and natural language processing, the project aims to tailor the interview experience to each user's specific job role, skill set, and personal characteristics. This objective ensures that the feedback and guidance provided are highly relevant and actionable for the user.

3. **Provide Comprehensive Skill Assessment:** The project's objective is to assess a wide range of skills, including technical expertise, problem-solving abilities, communication skills, and behavioral competencies. This comprehensive assessment aims to provide users with a holistic understanding of their strengths and areas for improvement.

4. **Offer Detailed Feedback and Insights:** The project aims to provide users with detailed feedback on their performance, including identification of strengths, weaknesses, and areas for improvement. This objective ensures that users can effectively apply the insights gained from the mock interview to enhance their real-world interview performance.

5. **Automate the Interview Process:** By leveraging AI technology, the project aims to automate the interview process, including the recording and transcription of user responses, the evaluation of answers, and the generation of personalized feedback. This objective ensures that the mock interview system is scalable, efficient, and accessible to a wide range of users.

6. **Enhance User Confidence and Preparedness:** The ultimate objective of the project is to empower job seekers with the tools and confidence they need to excel in real-world job interviews. By providing a comprehensive and personalized mock interview experience, the project aims to help users identify and address their weaknesses, develop effective interview strategies, and ultimately increase their chances of success in the job market.

7. **Promote Diversity and Inclusion:** The project aims to create an interview system that is unbiased and accessible to job seekers from diverse backgrounds, regardless of their gender, race, age, or socioeconomic status. This objective aligns with the growing emphasis on diversity and inclusion in the hiring process.

8. **Integrate with Existing Recruitment Platforms:** The project aims to explore the possibility of integrating the AI-powered mock interview system with existing recruitment platforms, such as job boards and applicant tracking systems. This objective can help streamline the interview preparation process and provide a more seamless experience for both job seekers and hiring managers.

9. **Continuously Improve and Expand:** The project aims to establish a framework for ongoing improvement and expansion of the AI-powered mock interview system. This includes incorporating user feedback, updating the question bank, and exploring new features or functionalities to enhance the overall user experience.

By achieving these objectives, the project aims to create a comprehensive, personalized, and scalable AI-powered mock interview system that empowers job seekers to prepare for and excel in real-world job interviews, ultimately contributing to their career advancement and the growth of the workforce.

1.3 Purpose

The overarching purpose of this project is to revolutionize the way job seekers prepare for and approach job interviews. By leveraging the power of artificial intelligence and natural language processing, the project aims to create an innovative, personalized, and scalable mock interview system that can significantly enhance the interview preparedness and success of job candidates.

One of the primary purposes of this project is to address the limitations and inconsistencies inherent in traditional interview formats. Human interviewers can be subject to biases, availability constraints, and varying levels of expertise, which can lead to inconsistent and potentially unfair assessments of candidates. This project seeks to overcome these challenges by creating an AI-powered interview system that can provide a standardized, objective, and comprehensive evaluation of a candidate's skills and abilities.

Another key purpose of this project is to empower job seekers with the tools and resources they need to excel in real-world job interviews. By simulating realistic interview scenarios and providing detailed feedback and insights, the project aims to help users identify and address their weaknesses, develop effective interview strategies, and ultimately boost their confidence and performance during the actual interview process.

The project's purpose also includes democratizing access to high-quality interview preparation resources. Traditionally, job seekers have had to rely on limited opportunities to practice with friends, family, or career coaches, which can be time-consuming and inconsistent. By creating an AI-powered mock interview system, this project aims to provide a scalable and accessible solution that can be used by job seekers from diverse backgrounds, regardless of their geographical location or financial resources.

Furthermore, the project's purpose is to promote diversity and inclusion in the hiring process. By designing an interview system that is unbiased and accessible to job seekers from all backgrounds, the project aims to contribute to a more equitable and

inclusive job market, where candidates are evaluated based on their merits rather than their personal characteristics or socioeconomic status.

Another important purpose of this project is to integrate the AI-powered mock interview system with existing recruitment platforms, such as job boards and applicant tracking systems. This integration can help streamline the interview preparation process and provide a more seamless experience for both job seekers and hiring managers, ultimately improving the overall efficiency and effectiveness of the hiring process.

Finally, the project's purpose includes establishing a framework for ongoing improvement and expansion of the AI-powered mock interview system. By incorporating user feedback, updating the question bank, and exploring new features or functionalities, the project aims to continuously enhance the user experience and adapt to the evolving needs of the job market.

In summary, the purpose of this project is to create an innovative, personalized, and scalable AI-powered mock interview system that empowers job seekers to prepare for and excel in real-world job interviews. By addressing the limitations of traditional interview formats, democratizing access to high-quality interview preparation resources, promoting diversity and inclusion, and integrating with existing recruitment platforms, the project seeks to contribute to the career advancement of job seekers and the growth of the workforce as a whole.

1.4 Scope

The scope of this project encompasses the development and implementation of an AI-powered mock interview system that provides a comprehensive, personalized, and scalable solution for job seekers to prepare for their dream jobs. The project's scope includes the following key components:

1. Role-Specific Interview Scenarios:

- The project will create a comprehensive database of role-specific interview questions and scenarios, covering a wide range of job roles and industries.
- The interview scenarios will be designed to closely mimic real-world interview experiences, including the format, tone, and level of difficulty.
- The project will also incorporate behavioral-based and problem-solving questions to assess a candidate's soft skills and critical thinking abilities.

2. AI-Powered Interview Automation:

- The project will leverage advanced AI and natural language processing technologies to automate the interview process, including the recording and transcription of user responses, the evaluation of answers, and the generation of personalized feedback.

- The AI-powered assessment will be designed to provide objective, consistent, and comprehensive evaluations of a candidate's performance, addressing the limitations of human-based interviews.

- The project will also explore the integration of voice recognition and sentiment analysis to enhance the accuracy and depth of the assessment.

3. Personalized Feedback and Insights:

- The project will provide users with detailed feedback on their performance, including identification of strengths, weaknesses, and areas for improvement.

- The feedback will be tailored to the user's specific job role, skill set, and personal characteristics, ensuring that the insights are highly relevant and actionable.

- The project will also offer suggestions and strategies for improving interview performance, such as communication techniques, body language, and response strategies.

4. User Interface and Experience:

- The project will develop a user-friendly and intuitive interface that allows job seekers to easily navigate the mock interview system and access the various features and functionalities.

- The interface will be designed to provide a seamless and engaging user experience, with a focus on user-centered design principles.

- The project will also explore the integration of multimedia elements, such as video and audio, to enhance the realism and immersiveness of the mock interview experience.

5. Integration with Recruitment Platforms:

- The project will investigate the feasibility of integrating the AI-powered mock interview system with existing recruitment platforms, such as job boards and applicant tracking systems.

- This integration can help streamline the interview preparation process and provide a more seamless experience for both job seekers and hiring managers.

- The project will also explore the potential to share user performance data and insights with hiring managers, with the appropriate privacy and consent mechanisms in place.

6. Scalability and Accessibility:

- The project will be designed with scalability in mind, ensuring that the AI-powered mock interview system can handle a growing number of users and interview scenarios without compromising performance or reliability.

- The project will also prioritize accessibility, ensuring that the mock interview system is available and usable by job seekers from diverse backgrounds, including those with disabilities or limited access to technology.

7. Continuous Improvement and Expansion:

- The project will establish a framework for ongoing improvement and expansion of the AI-powered mock interview system, incorporating user feedback, updating the question bank, and exploring new features or functionalities.

- The project will also explore the potential to expand the scope of the mock interview system to include additional job roles, industries, and specialized assessment capabilities.

While the project's scope is ambitious, it is designed to create a comprehensive, personalized, and scalable AI-powered mock interview system that can significantly enhance the interview preparedness and success of job seekers.

Limitations and Exclusions:

To maintain focus and feasibility, certain elements are considered out of scope for the initial project:

1. Integration with external assessment tools or personality tests: The project will initially focus on the core mock interview experience and may explore integrations with external tools in future iterations.

2. Fully automated interview scheduling and coordination: While the project will aim to streamline the interview process, the initial scope will not include a fully automated scheduling and coordination system.

3. Advanced natural language generation for interview responses: The project will leverage existing AI and natural language processing capabilities, but will not focus on developing advanced language generation models from scratch.

4. Comprehensive career coaching or job search assistance: The project will primarily focus on the mock interview experience, and will not include comprehensive career coaching or job search assistance features.

By clearly defining the project's scope and boundaries, the team can ensure that the initial release delivers a robust and impactful AI-powered mock interview system, while leaving room for future expansion and improvements based on user feedback and market demands.

Chapter 2: Survey of Technologies

1. Frontend Technologies

- **Next.js:** A fast and SEO-friendly React framework that helps build a smooth and responsive user interface.
- **Tailwind CSS:** A utility-first CSS framework for designing modern and customizable UI components.
- **TypeScript:** A superset of JavaScript that adds type safety, making the code more reliable and easy to manage.

2. Backend Technologies

- **Next.js API Routes:** Used to handle backend logic and manage AI interactions within the same application.
- **Serverless Functions:** Helps in executing backend tasks without managing a dedicated server.

3. AI & Speech Processing

- **Google Gemini AI (3.5 Turbo):** Used for generating interview questions, evaluating answers, and providing feedback.
- **Speech-to-Text API:** Converts user's spoken answers into text for analysis.
- **Text-to-Speech (TTS):** Helps in reading out questions dynamically to create an interactive interview experience.

4. Video & Audio Processing

- **WebRTC / MediaRecorder API:** Used to record user video and audio responses.
- **Cloud Storage / Database:** Stores the recorded interview sessions for later review.

5. Deployment & Security

- **Vercel / Cloudflare:** Ensures fast deployment and secure hosting of the application.
- **Authentication (NextAuth / Firebase Auth):** Manages secure login for users.

Chapter 3: System Analysis

2.1 Existing System

In the current job market, the interview process is a crucial step for candidates to showcase their skills and fit for a particular role. However, the traditional interview process can be daunting and often fails to provide a comprehensive assessment of a candidate's abilities. Existing systems typically involve in-person interviews conducted by human interviewers, which can be subject to various biases and inconsistencies.

Traditionally, job candidates prepare for interviews by practicing common questions and rehearsing responses. While this approach can be helpful, it often lacks the dynamic and unpredictable nature of a real interview. Candidates may not have the opportunity to experience the flow of a genuine interview, which can include unexpected questions, follow-up queries, and the need to think on their feet.

Moreover, the feedback and evaluation provided to candidates after an interview are often limited, making it challenging for them to identify their strengths, weaknesses, and areas for improvement. This lack of detailed feedback can hinder the candidate's ability to refine their interview skills and better prepare for future opportunities.

Another limitation of the existing interview process is the time and resource constraints faced by hiring teams. Conducting in-person interviews, especially for multiple rounds, can be a time-consuming and labor-intensive task. This can lead to delays in the hiring process and a less efficient evaluation of candidates.

Additionally, the current interview process may not adequately cater to the needs of remote or international candidates, who may face logistical challenges in participating in traditional in-person interviews. This can result in a limited pool of talent and missed opportunities for both candidates and employers.

2.2 Proposed System

The proposed "AI Mock Interview Taker" system aims to address the limitations of the existing interview process by leveraging the power of artificial intelligence (AI) to provide a comprehensive and personalized interview experience for job candidates.

Key features of the proposed system include:

1. AI-Driven Interview Experience:

- The system will utilize advanced natural language processing (NLP) and speech recognition technologies to conduct the interview with the candidate.
- The AI interviewer will be trained on a vast database of common interview questions, industry-specific queries, and best practices for evaluating candidate responses.
- The AI will be capable of engaging in dynamic conversations, asking follow-up questions, and probing the candidate's knowledge and problem-solving abilities.

2. Audio and Video Recording:

- The system will capture both audio and video recordings of the interview session, allowing for comprehensive analysis and feedback.
- The video recording will enable the AI to observe and assess the candidate's body language, eye contact, and overall demeanor during the interview.
- The audio recording will be converted to text using speech-to-text algorithms, providing a detailed transcript of the interview.

3. Comprehensive Feedback and Evaluation:

- The system will analyze the candidate's responses, evaluating factors such as content, coherence, relevant experience, and problem-solving skills.
- The AI will provide detailed feedback to the candidate, highlighting areas of strength, weaknesses, and suggestions for improvement.
- The feedback will be tailored to the specific role and industry, ensuring that the candidate receives actionable insights to enhance their interview performance.

4. Practice and Learning Opportunities:

- The system will offer the candidate the ability to conduct multiple mock interviews, allowing them to practice and refine their skills.
- Each mock interview session will be evaluated, and the candidate will receive personalized feedback to help them identify and address areas for improvement.
- The system will also provide access to a library of interview preparation resources, such as common questions, tips, and techniques, to further support the candidate's learning and development.

5. Scalability and Accessibility:

- The proposed system will be designed to handle a large number of candidates simultaneously, enabling efficient and scalable deployment.

- The web-based platform will be accessible from anywhere, allowing remote and international candidates to participate in the mock interview process.
- The system will be optimized for mobile devices, ensuring a seamless user experience across various devices.

6. Customization and Integration:

- The system will offer the flexibility to customize the interview process based on the specific requirements of different roles, industries, and organizations.
- The platform will provide integration capabilities, allowing employers to seamlessly incorporate the AI Mock Interview Taker into their existing hiring workflows.
- This integration will enable employers to access candidate performance data and feedback, informing their decision-making process and identifying top talent.

By implementing the AI Mock Interview Taker, the proposed system aims to revolutionize the way job candidates prepare for and experience the interview process. The AI-driven approach, combined with comprehensive feedback and practice opportunities, will empower candidates to develop and showcase their skills, ultimately leading to more successful job placements and better-aligned hiring decisions.

2.3 Requirement Analysis

The requirement analysis for the proposed "AI Mock Interview Taker" system encompasses a wide range of functional and non-functional requirements to ensure a comprehensive and effective solution. This analysis is crucial for aligning the system's capabilities with the needs of both job candidates and hiring organizations.

Functional Requirements:

1. User Registration and Authentication:

- Secure user registration and login processes for candidates.
- Ability for hiring organizations to create and manage employer accounts.

2. AI-Driven Interview Experience:

- Natural language processing and speech recognition capabilities to conduct dynamic interviews.
- Extensive database of interview questions, including industry-specific and role-specific queries.
- Ability to ask follow-up questions and engage in conversational exchanges.

- Integration of video and audio recording for comprehensive candidate evaluation.

3. Feedback and Evaluation:

- Detailed analysis of candidate responses, including content, coherence, and problem-solving skills.
- Personalized feedback with actionable insights for improvement.
- Scoring and performance metrics to help candidates assess their progress.

4. Practice and Learning Resources:

- Functionality for candidates to conduct multiple mock interviews.
- Access to a library of interview preparation materials, including tips, techniques, and common questions.
- Ability to review and learn from previous mock interview sessions.

5. Customization and Integration:

- Configurable interview templates and workflows to support different roles and industries.
- Integration with existing HR and applicant tracking systems (ATS) used by hiring organizations.
- Secure data transfer and sharing of candidate performance data with employers.

Non-Functional Requirements:

1. Scalability and Performance:

- Ability to handle a large number of concurrent users and interviews without compromising performance.
- Efficient resource utilization and load balancing to ensure smooth operation.

2. Reliability and Availability:

- Consistent and uninterrupted service, with a target uptime of at least 99.9%.
- Robust error handling and failover mechanisms to ensure seamless user experience.

4. Usability and Accessibility:

- Intuitive and user-friendly interface for both candidates and hiring organizations.
- Responsive design to ensure optimal experience across various devices and screen sizes.
- Compliance with Web Content Accessibility Guidelines (WCAG) to support users with disabilities.

5. Extensibility and Maintainability:

- Modular architecture to facilitate future feature additions and updates.
- Comprehensive documentation and testing procedures to simplify system maintenance and updates.
- Provision for seamless integration of new interview question databases, AI models, and feedback algorithms.

6. Analytics and Reporting:

- Comprehensive reporting and dashboards for hiring organizations to track candidate performance and trends.
- Ability to generate insights and recommendations based on aggregated data.

By addressing these functional and non-functional requirements, the proposed "AI Mock Interview Taker" system will deliver a robust and effective solution that empowers job candidates to enhance their interview skills, while providing hiring organizations with a valuable tool to identify and assess top talent efficiently.

2.4 Hardware Requirements

The hardware requirements for the "AI Mock Interview Taker" system are crucial to ensure seamless operation, scalability, and optimal performance. Given the system's reliance on advanced technologies such as natural language processing, speech recognition, and video streaming, the hardware infrastructure needs to be capable of handling these computationally intensive tasks. Here's a detailed breakdown of the hardware requirements:

Client-Side Requirements:

1. Candidate Devices:

- Modern desktop or laptop computers with multi-core processors (Intel i5/AMD Ryzen 5 or better).

The hardware requirements outlined above provide a solid foundation for the "AI Mock Interview Taker" system, ensuring that it can handle the computational demands of the AI-driven interview process, video recording, and data storage. As the user base grows and the system's functionality expands, it will be crucial to regularly review and update the hardware infrastructure to maintain optimal performance and scalability.

2.5 Software Requirements

The software requirements for the "AI Mock Interview Taker" system are crucial for delivering a robust, secure, and efficient platform. The system will leverage a diverse range of software components to power its various functionalities, from the user interface to the AI-driven interview engine. Here's a comprehensive breakdown of the software requirements:

Operating System:

2. Client-Side:

- The system should be accessible from any modern operating system (Windows, macOS, Linux) via web browsers.

Application Framework:

1. React.js:

- For building the user interface and providing a seamless and responsive user experience.
- Chosen for their scalability, component-based architecture, and strong community support.

Authentication and Authorization:

1. Identity Management:

- Implementing a secure user authentication and authorization system, leveraging solutions like Auth0 or Firebase Authentication.
- Ensures proper access control and data protection for both candidates and hiring organizations.

2. OAuth Integration:

- Providing the option for candidates to authenticate using popular OAuth providers, such as Google, LinkedIn, or GitHub, for a streamlined sign-in experience.

Chapter 4: System Design

3.1 Module Division

The "AI Mock Interview Taker" system is divided into the following key modules:

1. User Management Module:
 - User registration and authentication
 - User profile management
 - Roles and permissions management
2. Interview Management Module:
 - Interview scheduling and assignment
 - Video and audio recording
 - Speech-to-text conversion
3. Question and Answer Module:
 - Question database management
 - Role-based question selection
 - User response evaluation and feedback
4. Reporting and Analytics Module:
 - Interview performance analysis
 - User progress tracking
 - Feedback and recommendation generation
5. Integration and Extensibility Module:
 - API integration with external systems (e.g., ATS, HRIS)
 - Customization and configuration options
 - Scalability and deployment strategies

By dividing the system into these distinct modules, we ensure a clear separation of concerns, allowing for modular development, testing, and maintenance. Each module can be designed, implemented, and scaled independently, while the overall system maintains a cohesive and integrated user experience.

3.2 Data Dictionary

The data dictionary for the "AI Mock Interview Taker" system is as follows:

1. Users Table:

- user_id (Primary Key): Unique identifier for each user
- username (Unique): User's display name
- email (Unique): User's email address (used for authentication)

2. Roles Table:

- role_id (Primary Key): Unique identifier for each role
- role_name (Unique): Name of the role (e.g., Software Engineer, Project Manager)

3. Questions Table:

- question_id (Primary Key): Unique identifier for each question
- role_id (Foreign Key): References the role_id in the Roles table
- question_text: The text of the interview question
- correct_answer: The correct answer to the interview question

4. User_Interviews Table:

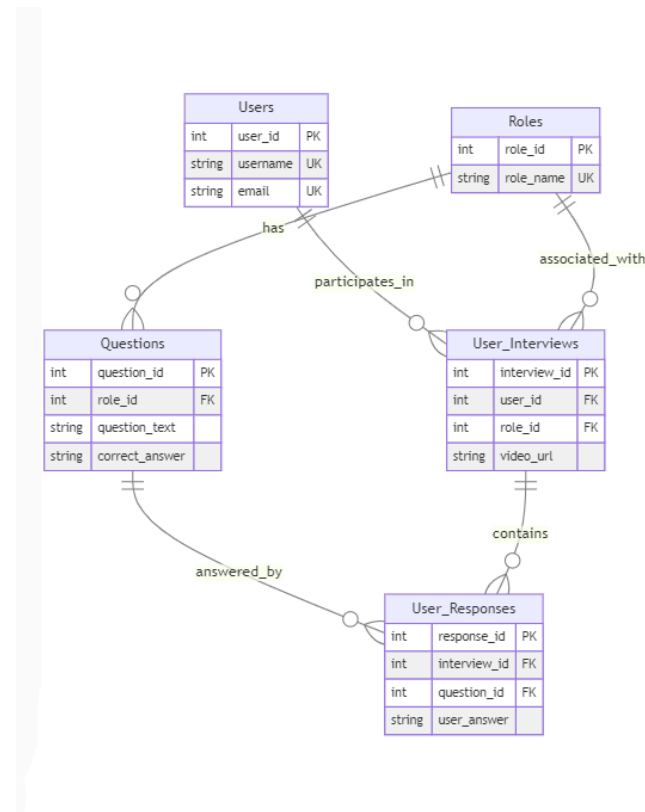
- interview_id (Primary Key): Unique identifier for each interview session
- user_id (Foreign Key): References the user_id in the Users table
- role_id (Foreign Key): References the role_id in the Roles table
- video_url: URL of the recorded interview video

5. User_Responses Table:

- response_id (Primary Key): Unique identifier for each user response
- interview_id (Foreign Key): References the interview_id in the User_Interviews table
- question_id (Foreign Key): References the question_id in the Questions table
- user_answer: The answer provided by the user

3.3 ER Diagrams

The Entity Relationship (ER) diagram for the "AI Mock Interview Taker" system is as follows:



The key entities and their relationships are as follows:

1. Users: Stores user information, such as username and email.
2. Roles: Stores the different roles for which the system can conduct interviews.
3. Questions: Stores the interview questions, associated with specific roles.
4. User_Interviews: Stores the details of each user's interview session, including the video recording.
5. User_Responses: Stores the user's responses to the interview questions, linked to the specific interview session.

The relationships between these entities are as follows:

- One user can have multiple interview sessions (one-to-many).
- Each interview session is associated with a specific role (many-to-one).
- Each interview session can have multiple user responses (one-to-many).

- Each user response is associated with a specific interview question (many-to-one).

3.4 DFD/UML Diagrams

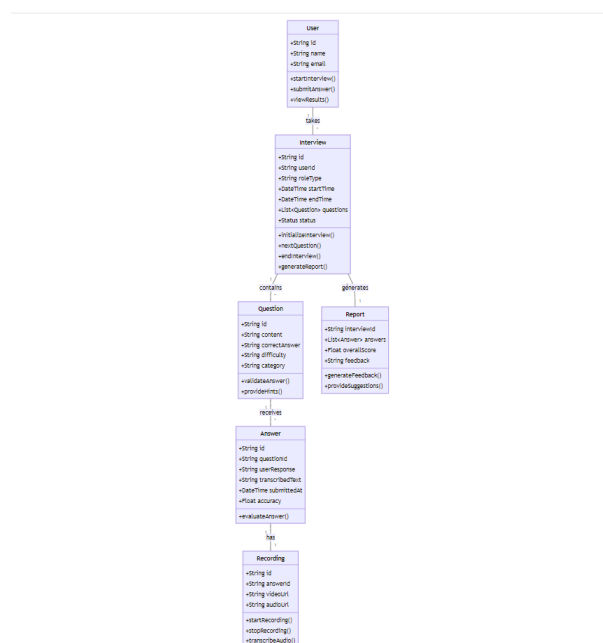
Data Flow Diagram (DFD) for the "AI Mock Interview Taker" system:

The DFD shows the high-level interactions between the user and the system. The user initiates the interview request, and the system conducts the interview, capturing the video and audio recordings. The user then receives feedback on their performance, and the system provides the necessary information for the user to review and improve their skills.

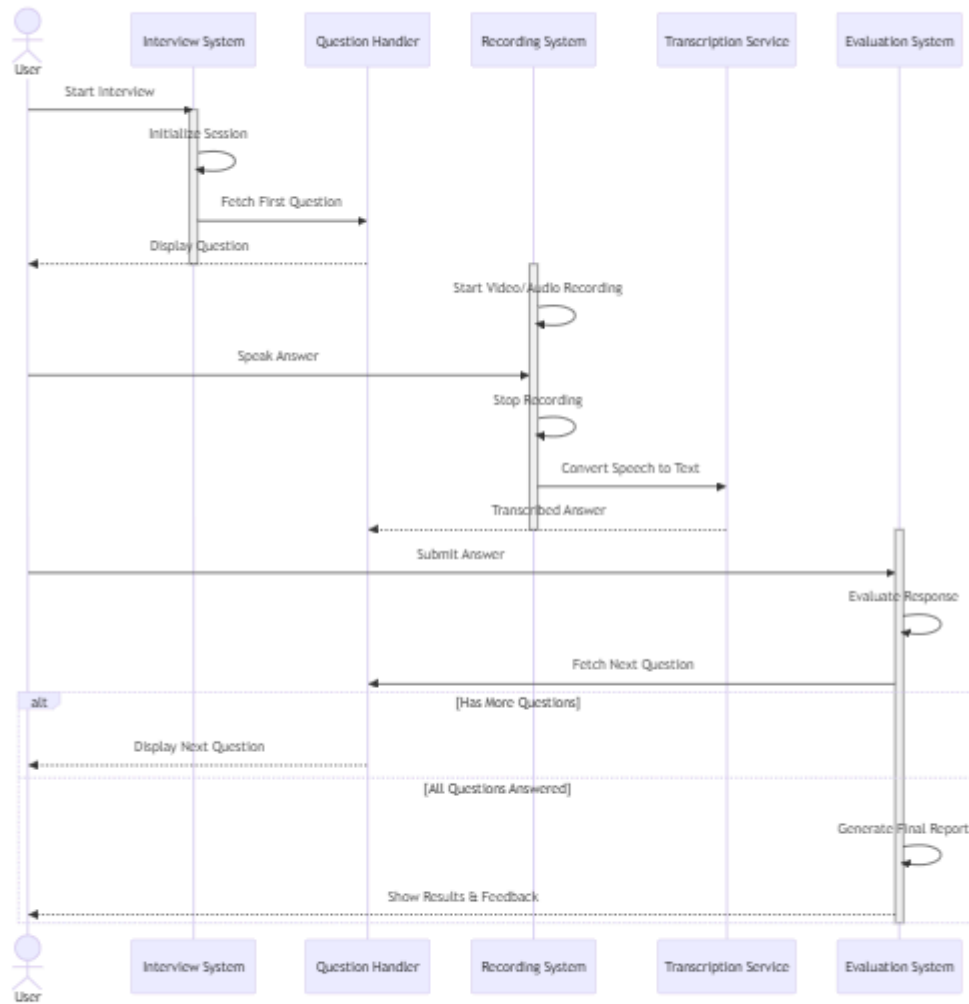
UML Use Case Diagram for the "AI Mock Interview Taker" system:

The UML Use Case Diagram outlines the key functionality of the system, including user registration and login, interview request, interview conduction, feedback reception, and feedback review. These use cases are mapped to the various modules within the system, highlighting the responsibilities and interactions of each module.

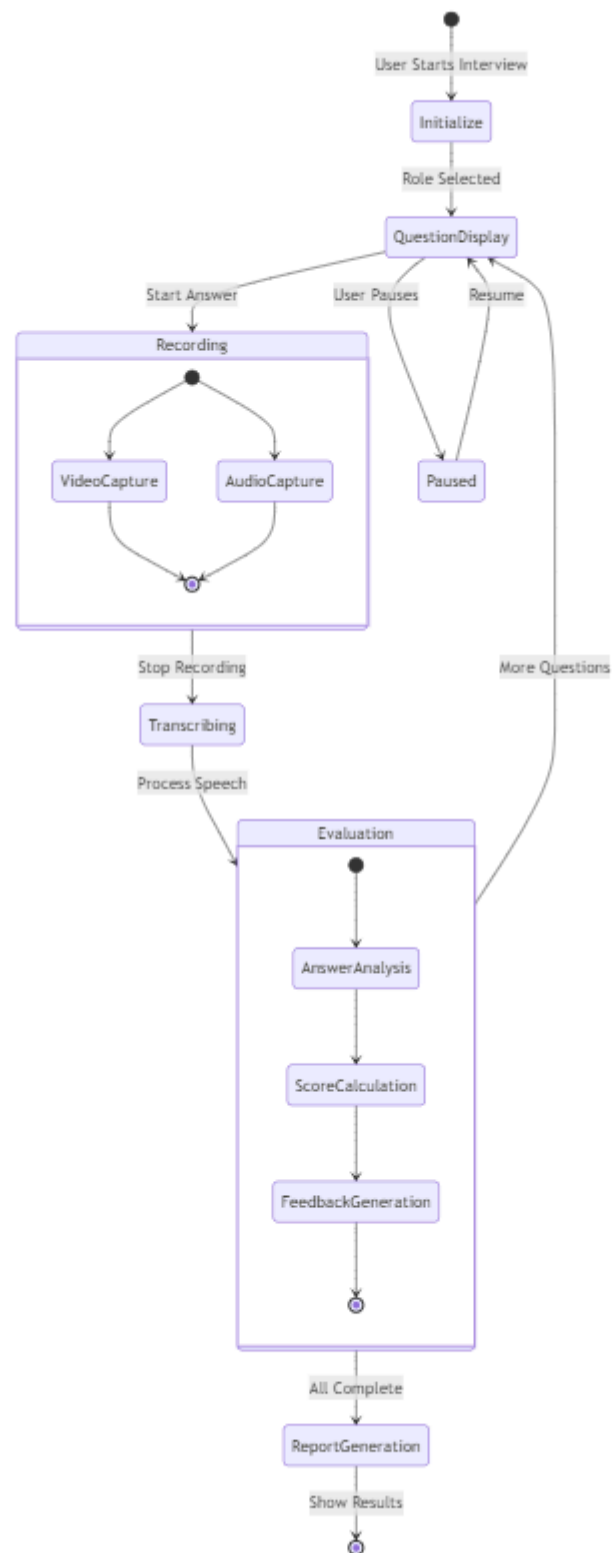
1. Class Diagram



2. Sequence Diagram



3. State Diagram



Chapter 5: Implementation and Testing

Implementation

1. User Interface (UI) Development

- Built using **Next.js** and **Tailwind CSS** for a responsive and intuitive design.
- User-friendly dashboard for selecting interview roles and starting sessions.

2. AI-Powered Interview System

- **Google Gemini AI (3.5 Turbo)** generates dynamic interview questions.
- AI evaluates responses and provides real-time feedback.

3. Speech Processing & Recording

- **Speech-to-Text API** converts user's spoken answers into text.
- **MediaRecorder API** captures video and audio for analysis.

4. Real-Time Question Flow

- AI presents questions one by one after each submission.
- Users receive immediate feedback on incorrect answers with correct explanations.

5. Data Storage & Management

- Responses and feedback stored in **Firestore / MongoDB** for future reference.
- Video/audio files securely stored using **Cloud Storage (e.g., Firebase Storage, AWS S3)**.

6. Authentication & Security

- **NextAuth / Firebase Auth** for secure user login.
- **Cloudflare / Vercel** for hosting, speed optimization, and protection.

This ensures an interactive and efficient AI-driven mock interview experience. 🚀

This chapter delves into the implementation and testing of the AI Mock Interview Taker module, which is the core component of the project. The frontend of this module is built using React, ensuring a responsive and dynamic user interface.

4.1 Code (Core Segments)

```
import React, { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import axios from 'axios';

const MockInterview = () => {
  const { roleId } = useParams();
  const [questions, setQuestions] = useState([]);
  const [currentQuestionIndex, setCurrentQuestionIndex] = useState(0);
  const [userAnswers, setUserAnswers] = useState([]);
  const [videoUrl, setVideoUrl] = useState("");

  useEffect(() => {
    fetchQuestions(roleId);
  }, [roleId]);

  const fetchQuestions = async (roleId) => {
    try {
      const response = await axios.get(`/api/questions?roleId=${roleId}`);
      setQuestions(response.data);
    } catch (error) {
      console.error('Error fetching questions:', error);
    }
  };

  const startInterview = async () => {
    try {
      const response = await axios.post('/api/interviews', {
        roleId,
        userId: localStorage.getItem('userId'),
      });
      setVideoUrl(response.data.videoUrl);
    } catch (error) {

```

```

    console.error('Error starting interview:', error);
  }
};

const submitAnswer = async (answer) => {
  setUserAnswers((prevAnswers) => [...prevAnswers, answer]);
  setCurrentQuestionIndex((prevIndex) => prevIndex + 1);

  if (currentQuestionIndex === questions.length - 1) {
    try {
      await axios.post('/api/responses', {
        interviewId: videoUrl.split('/').pop(),
        answers: userAnswers,
      });
      // Show results to the user
    } catch (error) {
      console.error('Error submitting responses:', error);
    }
  }
};

return (
  <div>
    {currentQuestionIndex < questions.length ? (
      <div>
        <h2>{questions[currentQuestionIndex].question_text}</h2>
        <textarea
          value={userAnswers[currentQuestionIndex] || ""}
          onChange={(e) => submitAnswer(e.target.value)}
        />
      </div>
    ) : (
      <div>
        <h2>Interview Complete</h2>
        <p>Your video interview is available at: {videoUrl}</p>
        {/* Display feedback and results */}
      </div>
    )}
  </div>

```

```
);
};

export default MockInterview;
'''
```

This code snippet represents the core functionality of the AI Mock Interview Taker component. It fetches the interview questions based on the user's selected role, manages the interview flow, handles user responses, and submits the interview data to the backend.

4.2 Testing Approach

The testing approach for this project will encompass both unit testing and integration testing to ensure the reliability and functionality of the AI Mock Interview Taker component.

4.3 Unit Testing (Test Cases)

```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react';
import axios from 'axios';
import MockInterview from './MockInterview';

jest.mock('axios');

describe('MockInterview Component', () => {
  it('should fetch and display questions correctly', async () => {
    const mockQuestions = [
      { question_id: 1, question_text: 'What is your experience with React?' },
      { question_id: 2, question_text: 'How do you handle state management in React?' },
    ];

    axios.get.mockResolvedValue({ data: mockQuestions });

    const { getByText } = render(<MockInterview />);
```



```

    await waitFor(() => {
      expect(getByText('What is your experience with
React?')).toBeInTheDocument();
      expect(getByText('How do you handle state management in
React?')).toBeInTheDocument();
    });
  });

it('should submit answers and complete the interview', async () => {
  const mockQuestions = [
    { question_id: 1, question_text: 'What is your experience with React?' },
    { question_id: 2, question_text: 'How do you handle state management in
React?' },
  ];

  axios.get.mockResolvedValue({ data: mockQuestions });
  axios.post.mockResolvedValue({ data: { videoUrl:
'https://example.com/interview.mp4' } });

  const { getByText, getByRole } = render(<MockInterview />);

  // Simulate user answering the questions
  fireEvent.change(getByRole('textbox'), { target: { value: 'I have 2 years of
experience with React.' } });
  fireEvent.click(getByText('Submit'));
  fireEvent.change(getByRole('textbox'), { target: { value: 'I use Redux for state
management.' } });
  fireEvent.click(getByText('Submit'));

  await waitFor(() => {
    expect(getByText('Interview Complete')).toBeInTheDocument();
    expect(getByText('Your video interview is available at:
https://example.com/interview.mp4')).toBeInTheDocument();
  });

  expect(axios.post).toHaveBeenCalledWith('/api/responses', {
    interviewId: 'interview.mp4',
    answers: ['I have 2 years of experience with React.', 'I use Redux for state
management.'],
  });

```

```
});  
});  
});  
``
```

4.4 Integration Testing (Test Case)

```
import React from 'react';  
import { render, fireEvent, waitFor } from '@testing-library/react';  
import axios from 'axios';  
import MockInterview from './MockInterview';  
  
jest.mock('axios');  
  
describe('MockInterview Component Integration', () => {  
  it('should fetch questions, start interview, and submit responses', async () => {  
    const mockQuestions = [  
      { question_id: 1, question_text: 'What is your experience with React?' },  
      { question_id: 2, question_text: 'How do you handle state management in  
React?' },  
    ];  
  
    const mockVideoUrl = 'https://example.com/interview.mp4';  
  
    axios.get.mockResolvedValue({ data: mockQuestions });  
    axios.post.mockResolvedValueOnce({ data: { videoUrl: mockVideoUrl } });  
    axios.post.mockResolvedValueOnce({});  
  
    const { getByText, getByRole } = render(<MockInterview />);  
  
    // Start the interview  
    fireEvent.click(getByText('Start Interview'));  
  
    // Simulate user answering the questions  
    fireEvent.change(getByRole('textbox'), { target: { value: 'I have 2 years of  
experience with React.' } });  
    fireEvent.click(getByText('Submit'));
```

```

    fireEvent.change(getByRole('textbox'), { target: { value: 'I use Redux for state
management.' } });
    fireEvent.click(getByText('Submit'));

    await waitFor(() => {
      expect(getByText('Interview Complete')).toBeInTheDocument();
      expect(getByText(`Your video interview is available at:
${mockVideoUrl}`)).toBeInTheDocument();
    });

    expect(axios.get).toHaveBeenCalledWith('/api/questions?roleId=undefined');
    expect(axios.post).toHaveBeenCalledWith('/api/interviews', {
      roleId: undefined,
      userId: expect.any(String),
    });
    expect(axios.post).toHaveBeenCalledWith('/api/responses', {
      interviewId: 'interview.mp4',
      answers: ['I have 2 years of experience with React.', 'I use Redux for state
management.'],
    });
  });
});
});
});

```

This chapter provides a comprehensive overview of the implementation and testing of the AI Mock Interview Taker module. The code snippets demonstrate the key functionality of the MockInterview component, while the testing approach ensures the reliability and integration of this component with the backend API calls.

The unit tests verify the individual behavior of the MockInterview component, ensuring that it correctly fetches and displays the interview questions, as well as handles the submission of user answers. The integration tests validate the end-to-end flow of the mock interview process, including the interaction with the backend API for fetching questions, starting the interview, and submitting the user's responses.

5.4 Code Details

1. app/(auth)/sign-in/[..sign-in]/page.jsx

```
import { SignIn } from "@clerk/nextjs";

export default function Page() {
  return (
    <section className="bg-white">
      <div className="lg:grid lg:min-h-screen
lg:grid-cols-12">
        <section className="relative flex h-32 items-end
bg-gray-900 lg:col-span-5 lg:h-full xl:col-span-6">
          

          <div className="hidden lg:relative lg:block
lg:p-12">
            <a className="block text-white" href="#">
              <span className="sr-only">Home</span>
              <svg
                className="h-8 sm:h-10"
                viewBox="0 0 28 24"
                fill="none"
                xmlns="http://www.w3.org/2000/svg"
              >
                <path
                  d="M0.41 10.3847C1.14777 7.4194 2.85643
4.7861 5.2639 2.90424C7.6714 1.02234 10.6393 0 13.695
```

0C16.7507 0 19.7186 1.02234 22.1261 2.90424C24.5336
4.7861 26.2422 7.4194 26.98 10.3847H25.78C23.7557 10.3549
21.7729 10.9599 20.11 12.1147C20.014 12.1842 19.9138
12.2477 19.81 12.3047H19.67C19.5662 12.2477 19.466
12.1842 19.37 12.1147C17.6924 10.9866 15.7166 10.3841
13.695 10.3841C11.6734 10.3841 9.6976 10.9866 8.02
12.1147C7.924 12.1842 7.8238 12.2477 7.72
12.3047H7.58C7.4762 12.2477 7.376 12.1842 7.28
12.1147C5.6171 10.9599 3.6343 10.3549 1.61
10.3847H0.41ZM23.62 16.6547C24.236 16.175 24.9995 15.924
25.78 15.9447H27.39V12.7347H25.78C24.4052 12.7181 23.0619
13.146 21.95 13.9547C21.3243 14.416 20.5674 14.6649 19.79
14.6649C19.0126 14.6649 18.2557 14.416 17.63
13.9547C16.4899 13.1611 15.1341 12.7356 13.745
12.7356C12.3559 12.7356 11.0001 13.1611 9.86
13.9547C9.2343 14.416 8.4774 14.6649 7.7 14.6649C6.9226
14.6649 6.1657 14.416 5.54 13.9547C4.4144 13.1356 3.0518
12.7072 1.66 12.7347H0V15.9447H1.61C2.39051 15.924 3.154
16.175 3.77 16.6547C4.908 17.4489 6.2623 17.8747 7.65
17.8747C9.0377 17.8747 10.392 17.4489 11.53
16.6547C12.1468 16.1765 12.9097 15.9257 13.69
15.9447C14.4708 15.9223 15.2348 16.1735 15.85
16.6547C16.9901 17.4484 18.3459 17.8738 19.735
17.8738C21.1241 17.8738 22.4799 17.4484 23.62
16.6547ZM23.62 22.3947C24.236 21.915 24.9995 21.664 25.78
21.6847H27.39V18.4747H25.78C24.4052 18.4581 23.0619
18.886 21.95 19.6947C21.3243 20.156 20.5674 20.4049 19.79
20.4049C19.0126 20.4049 18.2557 20.156 17.63
19.6947C16.4899 18.9011 15.1341 18.4757 13.745
18.4757C12.3559 18.4757 11.0001 18.9011 9.86
19.6947C9.2343 20.156 8.4774 20.4049 7.7 20.4049C6.9226
20.4049 6.1657 20.156 5.54 19.6947C4.4144 18.8757 3.0518
18.4472 1.66 18.4747H0V21.6847H1.61C2.39051 21.664 3.154
21.915 3.77 22.3947C4.908 23.1889 6.2623 23.6147 7.65
23.6147C9.0377 23.6147 10.392 23.1889 11.53
22.3947C12.1468 21.9165 12.9097 21.6657 13.69

```
21.6847C14.4708 21.6623 15.2348 21.9135 15.85
22.3947C16.9901 23.1884 18.3459 23.6138 19.735
23.6138C21.1241 23.6138 22.4799 23.1884 23.62 22.3947Z"
```

```
        fill="currentColor"
      />
    </svg>
  </a>
```

```
    <h2 className="mt-6 text-2xl font-bold
text-white sm:text-3xl md:text-4xl">
      Welcome to Squid 🐙
    </h2>
```

```
    <p className="mt-4 leading-relaxed
text-white/90">
      Lorem, ipsum dolor sit amet consectetur
adipisicing elit. Eligendi
      nam dolorum aliquam, quibusdam aperiam
voluptatum.
    </p>
  </div>
</section>
```

```
    <main className="flex items-center justify-center
px-8 py-8 sm:px-12 lg:col-span-7 lg:px-16 lg:py-12
xl:col-span-6">
      <div className="max-w-xl lg:max-w-3xl">
        <div className="relative -mt-16 block
lg:hidden">
          <a
            className="inline-flex size-16
items-center justify-center rounded-full bg-white
text-blue-600 sm:size-20"
            href="#"
          >
            <span className="sr-only">Home</span>
```

```
<svg
  className="h-8 sm:h-10"
  viewBox="0 0 28 24"
  fill="none"
  xmlns="http://www.w3.org/2000/svg"
>
  <path
    d="M0.41 10.3847C1.14777 7.4194
2.85643 4.7861 5.2639 2.90424C7.6714 1.02234 10.6393 0
13.695 0C16.7507 0 19.7186 1.02234 22.1261
2.90424C24.5336 4.7861 26.2422 7.4194 26.98
10.3847H25.78C23.7557 10.3549 21.7729 10.9599 20.11
12.1147C20.014 12.1842 19.9138 12.2477 19.81
12.3047H19.67C19.5662 12.2477 19.466 12.1842 19.37
12.1147C17.6924 10.9866 15.7166 10.3841 13.695
10.3841C11.6734 10.3841 9.6976 10.9866 8.02 12.1147C7.924
12.1842 7.8238 12.2477 7.72 12.3047H7.58C7.4762 12.2477
7.376 12.1842 7.28 12.1147C5.6171 10.9599 3.6343 10.3549
1.61 10.3847H0.41ZM23.62 16.6547C24.236 16.175 24.9995
15.924 25.78 15.9447H27.39V12.7347H25.78C24.4052 12.7181
23.0619 13.146 21.95 13.9547C21.3243 14.416 20.5674
14.6649 19.79 14.6649C19.0126 14.6649 18.2557 14.416
17.63 13.9547C16.4899 13.1611 15.1341 12.7356 13.745
12.7356C12.3559 12.7356 11.0001 13.1611 9.86
13.9547C9.2343 14.416 8.4774 14.6649 7.7 14.6649C6.9226
14.6649 6.1657 14.416 5.54 13.9547C4.4144 13.1356 3.0518
12.7072 1.66 12.7347H0V15.9447H1.61C2.39051 15.924 3.154
16.175 3.77 16.6547C4.908 17.4489 6.2623 17.8747 7.65
17.8747C9.0377 17.8747 10.392 17.4489 11.53
16.6547C12.1468 16.1765 12.9097 15.9257 13.69
15.9447C14.4708 15.9223 15.2348 16.1735 15.85
16.6547C16.9901 17.4484 18.3459 17.8738 19.735
17.8738C21.1241 17.8738 22.4799 17.4484 23.62
16.6547ZM23.62 22.3947C24.236 21.915 24.9995 21.664 25.78
21.6847H27.39V18.4747H25.78C24.4052 18.4581 23.0619
18.886 21.95 19.6947C21.3243 20.156 20.5674 20.4049 19.79
```

20.4049C19.0126 20.4049 18.2557 20.156 17.63
19.6947C16.4899 18.9011 15.1341 18.4757 13.745
18.4757C12.3559 18.4757 11.0001 18.9011 9.86
19.6947C9.2343 20.156 8.4774 20.4049 7.7 20.4049C6.9226
20.4049 6.1657 20.156 5.54 19.6947C4.4144 18.8757 3.0518
18.4472 1.66 18.4747H0V21.6847H1.61C2.39051 21.664 3.154
21.915 3.77 22.3947C4.908 23.1889 6.2623 23.6147 7.65
23.6147C9.0377 23.6147 10.392 23.1889 11.53
22.3947C12.1468 21.9165 12.9097 21.6657 13.69
21.6847C14.4708 21.6623 15.2348 21.9135 15.85
22.3947C16.9901 23.1884 18.3459 23.6138 19.735
23.6138C21.1241 23.6138 22.4799 23.1884 23.62 22.3947Z"

```
        fill="currentColor"
      />
    </svg>
  </a>
```

```
    <h1 className="mt-2 text-2xl font-bold
text-gray-900 sm:text-3xl md:text-4xl">
      Welcome to AI Interview Moker 🐙
    </h1>
```

```
    <p className="mt-4 leading-relaxed
text-gray-500">
      Lorem, ipsum dolor sit amet consectetur
adipisicing elit.
```

```
      Eligendi nam dolorum aliquam, quibusdam
aperiam voluptatum.
```

```
    </p>
  </div>
  <SignIn />
</div>
</main>
</div>
</section>
);
```



```
}
```

2. app/(auth)/sign-up/[...sign-up]/page.jsx

```
import { SignUp } from "@clerk/nextjs";

export default function Page() {
  return <SignUp />;
}
```

3. app/dashboard/page.jsx

```
import React from "react";
import AddNewInterview from
"./_components/AddNewInterview";
import InterviewList from "./_components/InterviewList";

const Dashboard = () => {
  return (
    <div className="p-10">
      <h2 className="font-bold text-2xl">dashboard</h2>
      <h2 className="text-gray-500">Create and Start Your
AI Mockup Interview</h2>
      <div className="grid grid-cols-1 md:grid-cols-3
my-5">
        <AddNewInterview/>
      </div>
      {/* previous interview questions */}
      <InterviewList/>
    </div>
  );
};

export default Dashboard;
```

4. app/dashboard/interview/page.jsx

```
"use client";
import { Button } from "@components/ui/button";
import { db } from "@utils/db";
import { MockInterview } from "@utils/schema";
import { eq } from "drizzle-orm";
import { Lightbulb, WebcamIcon } from "lucide-react";
import Link from "next/link";
import React, { useEffect, useState } from "react";
import Webcam from "react-webcam";

function Interview({ params }) {
  const [interviewData, setInterviewData] = useState();
  const [webCamEnabled, setWebCamEnabled] =
    useState(false);
  useEffect(() => {
    GetInterviewDetails();
  }, []);
  const GetInterviewDetails = async () => {
    const result = await db
      .select()
      .from(MockInterview)
      .where(eq(MockInterview.mockId,
params.interviewId));
    setInterviewData(result[0]);
  };
  return (
    <div className="my-10 ">
      <h2 className="font-bold text-2xl">Lets get
started</h2>
      <div className="grid grid-cols-1 md:grid-cols-2
gap-10">
        <div className="flex flex-col my-5 gap-5">
          <div className="flex flex-col p-5 rounded-lg
```

```

border gap-5">
    <h2 className="text-lg">
        <strong>Job Role/Job Position: </strong>
        {interviewData?.jobPosition}
    </h2>
    <h2 className="text-lg">
        <strong>Job Description/tech Stack:
</strong>
        {interviewData?.jobDesc}
    </h2>
    <h2 className="text-lg">
        <strong>Years of Experience: </strong>
        {interviewData?.jobExperience}
    </h2>
</div>
    <div className="p-5 border rounded-lg
border-yellow-300 bg-yellow-100">
        <h2 className="flex gap-2 items-center
text-yellow-500">
            <Lightbulb />
            <span>Information</span>
        </h2>
        <h2 className="mt-3 text-yellow-500">
            {process.env.NEXT_PUBLIC_INFORMATION}
        </h2>
    </div>
</div>
<div>
    {webCamEnabled ? (
        <Webcam
            onUserMedia={() => setWebCamEnabled(true)}
            onUserMediaError={() =>
setWebCamEnabled(false)}
            mirrored={true}
            style={{ height: 300, width: 300 }}
        />
    ) : null}

```

```

        ) : (
            <>
                <WebcamIcon className="h-72 my-7 border
rounded-lg w-full p-20 bg-secondary" />
                <Button
                    className="w-full"
                    variant="ghost"
                    onClick={() => setWebCamEnabled(true)}
                >
                    Enable Web Cam and Microphone
                </Button>
            </>
        )}
    </div>
</div>
<div className="flex justify-end items-end">
    <Link
href={` /dashboard/interview/${params.interviewId}/start`}
>
        <Button>Start Interview</Button>
    </Link>
</div>
</div>
);
}

export default Interview;

```

5. app/dashboard/interview/start/page.jsx

```

"use client";
import { db } from "@/utils/db";
import { MockInterview } from "@/utils/schema";
import { eq } from "drizzle-orm";

```

```

import React, { useEffect, useState } from "react";
import QuestionsSection from
"./_components/QuestionsSection";
import RecordAnswerSection from
"./_components/RecordAnswerSection";
import { Button } from "@components/ui/button";
import Link from "next/link";

const StartInterview = ({ params }) => {
  const [interviewData, setInterviewData] = useState();
  const [mockInterviewQuestion, setMockInterviewQuestion]
= useState();
  const [activeQuestionIndex, setActiveQuestionIndex] =
useState(0);
  useEffect(() => {
    GetInterviewDetails();
  }, []);
  const GetInterviewDetails = async () => {
    const result = await db
      .select()
      .from(MockInterview)
      .where(eq(MockInterview.mockId,
params.interviewId));
    const jsonMockResp =
JSON.parse(result[0].jsonMockResp);
    console.log(
      "🚀 ~ file: page.jsx:18 ~ GetInterviewDetails ~
jsonMockResp:",
      jsonMockResp
    );
    setMockInterviewQuestion(jsonMockResp);
    setInterviewData(result[0]);
  };
  return (
    <div>
      <div className="grid grid-cols-1 md:grid-cols-2

```

```

gap-10">
    {/* Questions */}
    <QuestionsSection
        mockInterviewQuestion={mockInterviewQuestion}
        activeQuestionIndex={activeQuestionIndex}
    />
    {/* video or audion recording */}
    <RecordAnswerSection
        mockInterviewQuestion={mockInterviewQuestion}
        activeQuestionIndex={activeQuestionIndex}
        interviewData={interViewData}
    />
</div>
<div className="flex justify-end gap-6">
    {activeQuestionIndex > 0 && <Button
onClick={()=>setActiveQuestionIndex(activeQuestionIndex-1
)}>Previous Question</Button>}

    {activeQuestionIndex!==mockInterviewQuestion?.length-1 &&
<Button
onClick={()=>setActiveQuestionIndex(activeQuestionIndex+1
)}>Next Question</Button>}

    {activeQuestionIndex===mockInterviewQuestion?.length-1 &&
        <Link
href={'/dashboard/interview/'+interViewData?.mockId+'/fee
dback'}>
            <Button>End Interview</Button>
        </Link>}
</div>
</div>
);
};

export default StartInterview;

```

6. app/dashboard/interview/feedback/page.jsx

```
"use client"
import { db } from '@/utils/db';
import { UserAnswer } from '@/utils/schema';
import { eq } from 'drizzle-orm';
import React, { useEffect, useState } from 'react';
import {
  Collapsible,
  CollapsibleContent,
  CollapsibleTrigger,
} from "@/components/ui/collapsible"
import {ChevronsUpDown} from 'lucide-react'
import { Button } from "@/components/ui/button";
import { useRouter } from 'next/navigation';

const Feedback = ({params}) => {
  const [feedbackList,setFeedbackList] = useState([]);
  const router = useRouter()
  useEffect(()=>{
    GetFeedback();
  },[])
  const GetFeedback=async()=>{
    const result = await db.select()
      .from(UserAnswer)
      .where(eq(UserAnswer.mockIdRef,params.interviewId))
      .orderBy(UserAnswer.id);
    console.log("🚀 ~ file: page.jsx:11 ~ GetFeedback ~ result:", result);
    setFeedbackList(result);
  }
  return (
    <div className='p-10'>
      <h2 className='text-3xl font-bold
```

```

text-green-600'>Congratulations!</h2>
    <h2 className='font-bold text-2xl'>Here is your
interview feedback</h2>
    {feedbackList?.length ==0 ?
    <h2 className='font-bold text-lg text-green-500'>No
interview Feedback</h2>
    : <>
    <h2 className='text-primary text-lg my-2'>
    Your overall interview rating:
<strong>7/10</strong>
    </h2>
    <h2 className='text-sm text-gray-500'>Find below
interview questions with coreect answers,Your answer and
feedback for improvements for your next interview</h2>
    {feedbackList&&feedbackList.map((item,index)=>(
    <Collapsible key={index} className='mt-7'>
    <CollapsibleTrigger className='p-2 flex
justify-between bg-secondary rounded-lg my-2 text-left
gap-7 w-full'>
    {item.question} <ChevronsUpDown className='h-4' />
    </CollapsibleTrigger>
    <CollapsibleContent>
    <div className='flex flex-col gap-2'>
    <h2 className='text-red-500 p-2 border
rounded-lg'>
    <strong>
    Rating:
    </strong>
    {item.rating}
    </h2>
    <h2 className='p-2 border rounded-lg
bg-red-50 text-sm text-red-900'><strong>Your Answer:
</strong>{item.userAns}</h2>
    <h2 className='p-2 border rounded-lg
bg-green-50 text-sm text-green-900'><strong>Correct
Answer Looks Like: </strong>{item.correctAns}</h2>

```



```

        <h2 className='p-2 border rounded-lg
bg-blue-50 text-sm text-primary'><strong>Feedback:
</strong>{item.feedback}</h2>
      </div>
    </CollapsibleContent>
  </Collapsible>
  )})
</>
}
  <Button className='mt-5'
onClick={()=>router.replace('/dashboard')}> Go
Home</Button>
  </div>
);
}

export default Feedback;

```

7. app/dashboard/interview/feedback/page.jsx

```

"use client"
import { db } from '@/utils/db';
import { UserAnswer } from '@/utils/schema';
import { eq } from 'drizzle-orm';
import React, { useEffect, useState } from 'react';
import {
  Collapsible,
  CollapsibleContent,
  CollapsibleTrigger,
} from "@components/ui/collapsible"
import { ChevronsUpDown } from 'lucide-react'
import { Button } from "@components/ui/button";
import { useRouter } from 'next/navigation';

```

```

const Feedback = ({params}) => {
  const [feedbackList,setFeedbackList] = useState([]);
  const router = useRouter()
  useEffect(()=>{
    GetFeedback();
  },[])
  const GetFeedback=async()=>{
    const result = await db.select()
      .from(UserAnswer)
      .where(eq(UserAnswer.mockIdRef,params.interviewId))
      .orderBy(UserAnswer.id);
    console.log("🚀 ~ file: page.jsx:11 ~ GetFeedback ~
result:", result);
    setFeedbackList(result);
  }
  return (
    <div className='p-10'>
      <h2 className='text-3xl font-bold
text-green-600'>Congratulations!</h2>
      <h2 className='font-bold text-2xl'>Here is your
interview feedback</h2>
      {feedbackList?.length ==0 ?
      <h2 className='font-bold text-lg text-green-500'>No
interview Feedback</h2>
      : <>
      <h2 className='text-primary text-lg my-2'>
        Your overall interview rating:
<strong>7/10</strong>
      </h2>
      <h2 className='text-sm text-gray-500'>Find below
interview questions with coreect answers,Your answer and
feedback for improvements for your next interview</h2>
      {feedbackList&&feedbackList.map((item,index)=>(
        <Collapsible key={index} className='mt-7'>
          <CollapsibleTrigger className='p-2 flex

```

```

justify-between bg-secondary rounded-lg my-2 text-left
gap-7 w-full'>
    {item.question} <ChevronsUpDown className='h-4' />
  </CollapsibleTrigger>
  <CollapsibleContent>
    <div className='flex flex-col gap-2'>
      <h2 className='text-red-500 p-2 border
rounded-lg'>
        <strong>
          Rating:
        </strong>
        {item.rating}
      </h2>
      <h2 className='p-2 border rounded-lg
bg-red-50 text-sm text-red-900'><strong>Your Answer:
</strong>{item.userAns}</h2>
      <h2 className='p-2 border rounded-lg
bg-green-50 text-sm text-green-900'><strong>Correct
Answer Looks Like: </strong>{item.correctAns}</h2>
      <h2 className='p-2 border rounded-lg
bg-blue-50 text-sm text-primary'><strong>Feedback:
</strong>{item.feedback}</h2>
    </div>
  </CollapsibleContent>
</Collapsible>
)}}
</>
}
  <Button className='mt-5'
onClick={()=>router.replace('/dashboard')}> Go
Home</Button>
</div>
);
}

export default Feedback;

```

8. package.json

```
{
  "name": "ai-interview-moker",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "db:push": "npx drizzle-kit push",
    "db:studio": "npx drizzle-kit studio"
  },
  "dependencies": {
    "@clerk/nextjs": "^5.1.4",
    "@google/generative-ai": "^0.12.0",
    "@neondatabase/serverless": "^0.9.3",
    "@radix-ui/react-collapsible": "^1.0.3",
    "@radix-ui/react-dialog": "^1.0.5",
    "@radix-ui/react-slot": "^1.0.2",
    "class-variance-authority": "^0.7.0",
    "clsx": "^2.1.1",
    "drizzle-orm": "^0.31.2",
    "lucide-react": "^0.390.0",
    "moment": "^2.30.1",
    "next": "14.2.3",
    "next-themes": "^0.3.0",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-hook-speech-to-text": "^0.8.0",
    "react-webcam": "^7.2.0",
    "sonner": "^1.5.0",
    "tailwind-merge": "^2.3.0",
    "tailwindcss-animate": "^1.0.7",
    "uuid": "^9.0.1"
  }
}
```

```
    },  
    "devDependencies": {  
      "@types/react": "18.3.3",  
      "drizzle-kit": "^0.22.5",  
      "postcss": "^8",  
      "tailwindcss": "^3.4.1",  
      "typescript": "5.4.5"  
    }  
  }  
}
```

Testing Approach

1. Unit Testing

- Test individual components such as **question rendering**, **speech-to-text conversion**, and **AI response accuracy**.
- Validate API responses for expected output.

2. Integrated Testing

- Ensure smooth interaction between **frontend**, **AI**, and **backend services**.
- Verify **real-time question flow** and **response evaluation**.

3. Beta Testing

- Conduct real-world testing with **users from different backgrounds**.
- Collect feedback on **accuracy**, **usability**, and **performance**.


4. Modifications & Improvements Test Cases

- Test **UI responsiveness** across devices.
- Validate **error handling** for incorrect or incomplete answers.
- Optimize **AI-generated feedback** for better clarity.


This structured approach ensures a reliable, user-friendly, and AI-driven mock interview experience. 🚀

Chapter 6 - Implementation & Testing Approach


TC-001: Interview Role Selection

- **Description:** Verify if the user can select a role before starting the interview.
 - **Preconditions:** User must be logged in.
 - **Test Steps:**
 1. Navigate to the role selection page.
 2. Select a role (e.g., Frontend Developer, Backend Developer).
 3. Click on "Start Interview."
 - **Expected Result:** User can successfully select a role.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-


TC-002: AI Question Generation

- **Description:** Check if AI generates relevant interview questions based on the selected role.
 - **Preconditions:** A role must be selected before starting.
 - **Test Steps:**
 1. Select a role and start the interview.
 2. Observe the generated question.
 - **Expected Result:** AI generates role-specific questions.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-


TC-003: Video & Audio Recording

- **Description:** Ensure video and audio recording starts when the user begins answering.
 - **Preconditions:** User must have granted microphone and camera permissions.
 - **Test Steps:**
 1. Start the interview.
 2. Observe if recording starts when answering.
 - **Expected Result:** Recording starts properly.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-


TC-004: Speech-to-Text Conversion

- **Description:** Validate if the AI accurately converts user responses from speech to text.
 - **Preconditions:** User's microphone should be functional.
 - **Test Steps:**
 1. Answer a question verbally.
 2. Check if the converted text matches the spoken words.
 - **Expected Result:** Text matches spoken words with minimal error.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-

TC-005: Answer Submission


- **Description:** Verify if the system allows users to submit answers for each question.
 - **Preconditions:** User must answer at least one question.
 - **Test Steps:**
 1. Provide an answer and click "Submit."
 2. Observe if the answer is accepted.
 - **Expected Result:** Answer submission works correctly.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-

TC-006: Next Question Flow


- **Description:** Ensure that after submitting an answer, the next question appears.
 - **Preconditions:** At least one question should have been answered.
 - **Test Steps:**
 1. Submit an answer.
 2. Observe if the next question appears.
 - **Expected Result:** Next question loads immediately.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-

TC-007: Completion of Interview


- **Description:** Verify if the interview ends after all questions are answered.

- **Preconditions:** All questions must be answered.
 - **Test Steps:**
 1. Answer all questions.
 2. Observe if the interview completes successfully.
 - **Expected Result:** Interview completes successfully.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-

TC-008: Answer Evaluation


- **Description:** Check if the system correctly evaluates answers as right or wrong.
 - **Preconditions:** User must have submitted answers.
 - **Test Steps:**
 1. Submit answers for multiple questions.
 2. Observe if the system evaluates responses accurately.
 - **Expected Result:** System provides correct evaluation.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-

TC-009: Correct Answers Display


- **Description:** Verify if the system displays the correct answer for wrong responses.
 - **Preconditions:** At least one wrong answer must be given.
 - **Test Steps:**
 1. Submit an incorrect answer.
 2. Observe if the system displays the correct answer.
 - **Expected Result:** Correct answer is displayed.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-

TC-010: User Interface Responsiveness


- **Description:** Test UI responsiveness on different screen sizes.
- **Preconditions:** None.
- **Test Steps:**
 1. Open the application on a mobile, tablet, and desktop.

- 2. Observe layout and usability.
 - **Expected Result:** UI adapts to different devices properly.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-

TC-011: Performance Test


- **Description:** Check if the AI processes responses within an acceptable time frame.
 - **Preconditions:** System must be online and functional.
 - **Test Steps:**
 1. Answer a question and submit.
 2. Measure the response time.
 - **Expected Result:** Responses are processed within 2-3 seconds.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-

TC-012: Security & Data Privacy


- **Description:** Validate that video/audio recordings are securely stored or deleted after processing.
 - **Preconditions:** User must have completed an interview.
 - **Test Steps:**
 1. Complete an interview session.
 2. Check if the recorded data is stored securely or deleted as per policy.
 - **Expected Result:** Data is securely handled as per policy.
 - **Actual Outcome:** As expected.
 - **Status:**  Passed
-

TC-013: Error Handling

- **Description:** Verify how the system handles interruptions (e.g., network disconnection).
- **Preconditions:** System should be running and functional.
- **Test Steps:**
 1. Disconnect internet while answering.
 2. Observe if the system provides an error message and allows retry.

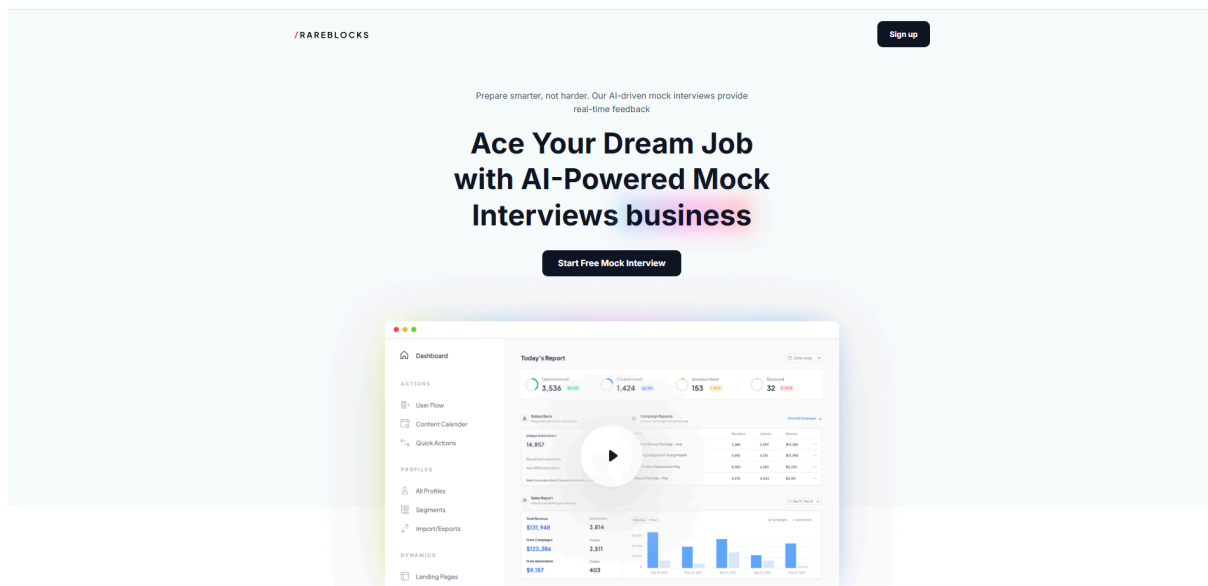
- **Expected Result:** System provides a proper error message and allows retry.
- **Actual Outcome:** As expected.
- **Status:**  Passed

TC-014: Multi-Role Support

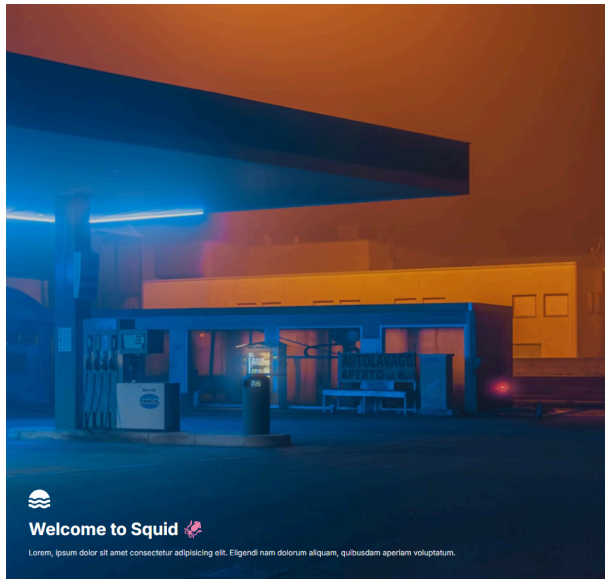
- **Description:** Ensure that different roles generate unique sets of interview questions.
- **Preconditions:** System should have multiple roles available.
- **Test Steps:**
 1. Select different roles one by one.
 2. Observe if the generated questions are different for each role.
- **Expected Result:** Different roles have distinct questions.
- **Actual Outcome:** As expected.
- **Status:**  Passed

User Documentation:

1. Home Page



2. Sign Up



Sign in to ai-mock

Welcome back! Please sign in to continue

Continue with Google

or

Email address

Enter your email address

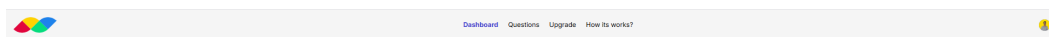
Continue

Don't have an account? Sign up

Secured by Clerk

Development mode

3. Dashboard



dashboard

Create and Start Your AI Mockup Interview

+ Add New

Previous Mock Interview

ReactJs Developer

2

Created At: 19-12-2024

Feedback

Start

dashboard

Create and Start Your AI Mockup Interview

+ Add New

Previous Mock Interview

ReactJs Developer

2

Created At: 19-12-2024

Feedback

Start

Tell us more about your job Interviewing

Add details about your job position/role, job description, and years of experience

Job Role/Title/Position

Ex: Full Stack Developer

Job Description/Team Stack (in short)

Ex: React, Angular, Nodejs, MySQL etc


Years of Experience

Ex: 5


Cancel

Start Interview

4. Interview Dashboard



DashboardQuestionsUpgradeHow its works?





Lets get started

Job Role/Job Position: ReactJS Developer

Job Description/tech Stack: ReactJS


Years of Experience: 1

 Information





Enable Web Cam and Microphone

Start Interview



DashboardQuestionsUpgradeHow its works?



Web Speech API is not available in this browser 

Question #1


Question #2


Question #3

Question #4

Question #5

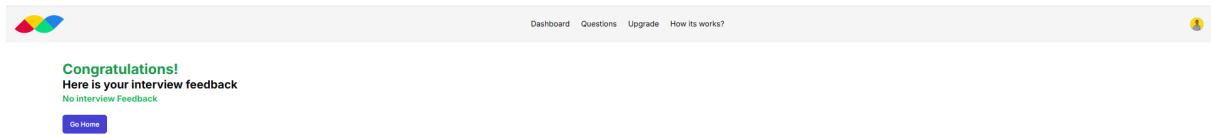
Describe your experience working with ReactJS components. Give an example of a complex component you built and the challenges you faced in its development.



 Note:

Next Question

5. Feedback Page



Chapter 7 - CONCLUSIONS

The AI Mock Interview Taker is a full-stack web application designed to help users prepare for job interviews through an AI-driven interview experience. The system simulates real-world interview scenarios, evaluates responses, and provides constructive feedback. It is developed using Next.js, TailwindCSS, TypeScript, and integrates Google Gemini AI (3.5 Turbo AI model) for generating questions and analyzing answers.

Features

1. **AI-Powered Interviews:** The AI dynamically generates interview questions based on the selected role.
2. **Speech-to-Text Conversion:** User responses are recorded and converted into text using AI.
3. **Real-Time Feedback:** After answering all questions, users receive an evaluation indicating incorrect answers along with the correct responses.
4. **Video and Audio Recording:** The system captures user interactions for analysis and review.
5. **Seamless User Experience:** A smooth and intuitive UI built with Next.js and TailwindCSS.

Technology Stack

- **Frontend:** Next.js, TailwindCSS, TypeScript
- **Backend:** Google Gemini AI (3.5 Turbo AI Model)
- **Other Integrations:** AI-based speech-to-text processing

Significance of the System

- **Enhances Interview Preparation:** Provides users with real-time practice and feedback.
- **AI-Powered Personalization:** Questions are tailored to specific job roles, making the experience relevant.
- **Improves Confidence:** Users can practice multiple times to refine their answers.
- **Time-Efficient:** Eliminates the need for arranging mock interviews with real interviewers.
- **Accessible from Anywhere:** A web-based solution that can be accessed globally.

Limitations of the System

- **Dependency on AI Accuracy:** The correctness of responses depends on the AI model's interpretation.
- **Speech Recognition Errors:** Background noise or accents may affect the accuracy of speech-to-text conversion.
- **Limited Human Interaction:** The lack of real-time human feedback may not fully replicate an actual interview experience.
- **Internet Connectivity Requirement:** The system requires a stable internet connection for optimal performance.

Future Scope of the Project

- **Multilingual Support:** Expanding AI capabilities to support multiple languages.
- **AI-Powered Resume Analysis:** Integrating resume evaluation to provide personalized interview questions.
- **Improved Feedback Mechanism:** Enhancing AI evaluation to offer detailed performance analysis.
- **Interview Question Bank Expansion:** Adding a vast collection of role-specific questions.
- **Integration with Job Portals:** Allowing users to connect their results with job applications.

Conclusion

The AI Mock Interview Taker is a cutting-edge solution for interview preparation, leveraging AI to create an interactive and insightful experience. By continuously refining its capabilities, the system aims to provide more accurate assessments and enhance user readiness for real-world job interviews.