A

PROJECT REPORT ON

## “JARVIS”

For

## NeoSoft Services

BY

## Manish Jangid Seat No. :10791

UNDER THE GUIDANCE OF

## Ms. Kalyani Sahastrabuddhe

SAVITRIBAI PHULE PUNE UNIVERSITY (SPPU) IN PARTIAL FULFILLMENT OF DEGREE FOR MASTER OF COMPUTER APPLICATIONS

Dr. D. Y. PATIL UNITECH SOCIETY’S

Dr. D. Y. PATIL INSTITUTE OF MANAGEMENT AND RESEARCH, PIMPRI, PUNE-18

2023 - 2025

Recipient of the "Best College Award" of Savitribai Phule Pune University

Accredited by NAAC with "A++ Grade” (CGPA 3.52)

MBA & MCA Programme Accredited by NBA



**CERTIFICATE**

This is to certify that **Manish Tulcharam Jangid** has successfully completed the project on **‘‘JARVIS ’’**as a partial fulfillment of his **Master of Computer Applications (MCA)** under the curriculum of **Savitribai Phule Pune University, Pune** for the academic year 2024-25

Ms.Kalyani Sahastrabuddhe Dr. Shikha Dubey Dr. Vishal Wadajkar

### Project Guide H.O.D. MCA Director

Signature Signature

Name Name

### Internal Examiner External Examiner

Date : Date :



INTERNSHIP COMPLETION CERTIFICATE

Dear Mr. **Manish Jangid Date: 24th Apr 2025**

This is to certify that **Mr. Manish Tulcharam Jangid** pursuing MCA from Dr. D.Y. Patil Institute of Management and Research, Pune has completed internship with **NeoSoft Services**. He has been working on the project Title : "JARVIS" during this period and also completed the tasks assigned to him as "Software Developer Intern".

The Internship started on **7th January 2025**. The intern has completed the assigned tasks within the mentioned period. The technologies used in the project are as listed below: **Frontend:** Next.js (React-based framework)

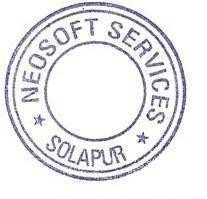
**Database:** PostgreSQL (via Drizzle ORM)

**API Models:** Gemini-2.0-Flash API

We wish him all the Best.

Regards,

**For Neosoft Services, Solapur**

****

**Director**

### ACKNOWLEDGEMENT

The successful completion of my final-year project, **"JARVIS – Job AI Ready Virtual Interview System"**, marks a significant milestone in my academic journey, and I would like to express my heartfelt gratitude to all those who supported and guided me throughout this endeavor.

First and foremost, I am profoundly thankful to **Dr. Vishal Wadajkar (Director)**, and **Dr. Shikha Dubey (Head of Department)** for providing an environment that fosters innovation and research. Their leadership and vision encouraged me to aim higher and push the boundaries of what I could achieve.

I extend my deepest appreciation to my project guide, **Ms. Kalyani sahastrabuddhe**, for her continuous support, insightful feedback, and expert guidance. Her valuable suggestions and mentorship were instrumental in shaping the project and helping me navigate technical challenges with clarity and confidence.

I would also like to thank all the faculty members for their encouragement and for sharing their knowledge, which has been a source of constant learning. A special mention goes to the laboratory staff for their timely help and cooperation during the practical implementation of the system.

A sincere thank you to my friends and fellow students who stood by me during every phase of this journey. Their honest feedback and camaraderie made this process enjoyable and enriching.

Finally, I owe everything to my family, whose unwavering support, patience, and belief in me have been my greatest strength. Their motivation kept me going even in the toughest moments. This project would not have been possible without the collective support of each of these individuals, and I am genuinely grateful to have had them by my side.

Yours Sincerely, Manish Jangid

### DECLARATION

This declaration is part of the project work entitled **“JARVIS”** is submitted as part of academic requirement for 4th semester of MCA Management.

I Manish Tulcharam Jangid solely declare that

1. I have not used any unfair means to complete the project.
2. I have followed the discipline and the rules of the organization where I was doing the project.
3. I have not been part of any act which may impact the college reputation adversely. The information I have given is true, complete and accurate. I understand that failure to give truthful, incomplete and inaccurate information may result in cancellation of my project work.

Yours Sincerely, Manish Jangid

**TABLE OF CONTENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter** |  | **Title** | **Page No** |
| **1** |  | **Introduction** | 1-6 |
|  | 1.1 | Abstract | 2 |
|  | 1.2 | Company Profile | 2-3 |
|  | 1.3 | Existing System and Need for System | 3 |
|  | 1.4 | Scope of System | 3-4 |
|  | 1.5 | Operating Environment - Hardware and Software | 4 |
|  | 1.6 | Brief Description of Technology Used | 4-6 |
| **2** |  | **Proposed System** | 7-10 |
|  | 2.1 | Proposed System | 8 |
|  | 2.2 | Feasibility Study | 8-9 |
|  | 2.3 | Objectives of Proposed System | 9 |
|  | 2.4 | Modules of Proposed System | 10 |
| **3** |  | **Analysis and Design** | 11-26 |
|  | 3.1 | System Requirements | 12-13 |
|  | 3.2 | Entity Relationship Diagram (ERD) | 11 |
|  | 3.3 | Table Structure | 14 |
|  | 3.4 | Use Case Diagram | 15-16 |
|  | 3.5 | Class Diagram | 17 |
|  | 3.6 | Activity Diagram | 18 |
|  | 3.7 | Sequence Diagram | 19 |
|  | 3.8 | Deployment Diagram | 20 |
|  | 3.9 | Module Hierarchy Diagram | 21 |
|  | 3.10 | Sample Input and Output Screens | 22-26 |
| **4** |  | **Coding** | 27-49 |
|  | 4.1 | Code snippets | 28-49 |
| **5** |  | **Testing** | 50-52 |
|  | 5.1 | Test Strategy | 51 |
|  | 5.2 | Test Cases | 52 |
| **6** |  | **Limitations of Proposed System** | 53-54 |
| **7** |  | **Proposed Enhancements** | 55-57 |
| **8** |  | **Conclusion** | 58-60 |
| **9** |  | **Bibliography** | 61-63 |
| **10** |  | **User Manual** | 64-73 |

# Chapter 1: Introduction

## Chapter1:Introduction

### Abstract

In today’s competitive job market, performing well in interviewsis a key factor for success. However, not everyone has access to regular mock interviews, professional mentors, or structured practice environments. JARVIS (Job AIReady Virtual Interview System) is a web- based, AI- powered mock interview platform that addresses this gap by offering a smart, automated, and personalized interview simulation system.

This project uses modern technologies like **Next.js**, **PostgreSQL**, and **Drizzle ORM**, combined with AI tools such as **GPT APIs**, to deliver domain-specific questions, evaluate responses, and provide detailed feedback. The system supports role-based question sets, user authentication, performance tracking, and a dashboard-driven experience. It is designed to help students, job seekers, and institutions conduct interview preparation efficiently and effectively.

JARVIS empowers candidates to rehearseinterviews, reduce anxiety, and improve their skills through consistent and intelligent feedback, making it a valuable solution for placement readiness and career growth.

### Company Profile

This project was developed under the guidance and mentorship of Neosoft Services, a recognized software solutions provider established in January 2011. Neosoft Services specializes in Web, Desktop, and Hybrid Mobile Application Development, focusing on innovation, timely delivery, and customer satisfaction.

With a presence in the ever-evolving global tech landscape, Neosoft Services standsout by delivering highly distinct digital identities for its clients. The company prides itself on its innovative vision and standard of excellence, which drives every project — whether for local enterprises or global brands.

The organization aims to make technology a true asset for its clients by thoroughly evaluating business needs and offering robust, scalable, and high-quality software solutions. Their customer- centric approach ensures that every product not only meets but often exceeds expectations in terms of usability, performance, and visual impact.

By integrating modern technologies and following industry best practices, Neosoft Services has built a reputation for delivering innovative solutions.

**Website:** <https://neosoftservices.co.in/>

### Existing System and Need for System

Traditional mock interview systems aretypically:

* + - **Manual and Time-Consuming**: Depend heavily on human evaluators.
    - **Subjective**: Feedback is not always consistent or data-driven.
    - **Generic**: Often lack role-based customization.
    - **Limitedin Analytics**: Donot provide measurableinsights.

Theselimitations make it difficult for candidates to identify weakareas or track progress.

**JARVIS** solves these problems by automatingthe entire process — from dynamic question generation to feedback delivery. It offers scalability, personalization, and real-time evaluation using AI models, thereby making interview preparation more accessible and effective.

### Scope of System

#### JARVIS is built for:

* + - **Students** preparing for campus placements.
    - **Job seekers** who want regularmock practice.
    - **Institutions** that want to offer interview trainingat scale. It

#### provides a digital platform where users can:

* + - Attempt role-specific mock interviews.
    - Record and store responses.
    - Receive intelligent, structured feedback.

#### Out of Scope:

* + - Real-timehuman interviewer interactions.
    - Advanced biometric features (e.g., emotion detection, eye tracking).
    - Resume based Questions (planned as a future enhancement).

### Operating Environment – Hardware and Software

JARVIS is a web-based platform that can be used across multiple devices and platforms. It uses the following hardware and software configuration:

* + - **Operating Systems**: Windows 10/11, Linux (Ubuntu), macOS Ventura+
    - **Browser Compatibility**: Chrome, Firefox, Edge, Safari
    - **Database**: PostgreSQL (relational database), managed using Drizzle ORM
    - **Backend Stack**: JavaScript (Next.js API routes), Drizzlefor ORM
    - **Frontend Stack**: React(via Next.js), Tailwind CSS forstyling

No special hardware is required beyond standard computing devices with internet access and a webcam/microphone for future features.

### Brief Description of Technology Used

The JARVIS mock interview platform is built using a full-stack web development architecture that combines a modern frontend, a serverless backend, AI-powered APIs, and a cloud-based database. Below is a detailed breakdown of the actual technologies used:

#### Next.js(Frontend and API Layer)

JARVISis builtusing **Next.js**, a React-basedframeworkthatpowers both the **frontend interface** and **API routes**. It enables server-side rendering, dynamic routing, and fast performance. Pages like Sign In, Dashboard, Interview Session, and Feedback are all developed using this framework.

#### React.js(UI Components)

React components are used to create dynamic and interactive UI elements such as:

* + Interviewquestion cards
  + Collapsible feedback per question
  + Responsive dashboards

#### Tailwind CSS (Styling)

Tailwind CSS is used for consistent and responsive styling across the platform. It helps create a clean and professional UI for all components including buttons, cards, modals, and feedback sections.

#### React Speech Recognition (Speech-to-Text Hook)

JARVIS integrates react-speech-recognition, a library that uses the browser’s Web Speech API to convert spoken responses into text. This allows users to answer interview questions via **voice input**, improving realism and accessibility.

#### Gemini Flash 2.0 API (AI Integration)

Thesystem uses **Gemini Flash 2.0**, a powerfullargelanguagemodel from Google, integrated via API. It performs two key functions:

* + **Question Generation**: It dynamically generatesinterview questionsbased on selected roles like “Frontend Developer” or “HR”.
  + **Answer Analysis**: It analyzes each answer submitted by the user and returns intelligent, structured feedbackfocusingon **clarity, fluency, tone, and relevance**.

This AI integration removes theneed for manually curated question banks and provides personalized feedback at scale.

#### Drizzle ORM (Database Layer)

The backend of JARVIS uses **Drizzle ORM**, a modern TypeScript-based ORM to handle interaction with the database. It provides type-safe schema definitions, smooth query building, and easy migration support.

#### PostgreSQL (Database)

Data such as users, interview sessions, questions, answers, and feedback are stored in a **Neon serverless PostgreSQL database**, making the system scalable and reliable. It allows efficient retrieval and storage of session data and user activity.

Thistech stack ensures JARVISis:

* + **AI-powered**
  + **Speech-enabled**
  + **Fully responsive**
  + **Serverless and scalable**

It creates a realistic, engaging, and professional mock interview experience for users preparing for job interviews.

# Chapter 2: Proposed System

## Chapter 2: Proposed System

### Proposed System

Theproposedsystem, **JARVIS (Job AI Ready Virtual Interview System)**, isan AI-powered mock interview platform designed to simulate real interview environments using modern web technologies. The platform addresses the limitations of traditional mock interview methods by offeringa fully digital, voice-enabled, and intelligent solution for self-practice.

Users can sign up, attempt role-specificinterviews, answer questions by speaking, and receive structured AI feedback. Unlike manual systems, JARVIS offers personalization, instant analysis, and performance tracking, all within an intuitive web interface.

The system is suitablefor students preparing forplacements, professionals preparing for interviews, and institutions wanting to offer mock interview solutions to large batches.

### Feasibility Study

Before development, a detailed feasibility study was conducted across three main aspects:

#### Technical Feasibility

* + Thesystem uses **Next.js** forbuilding scalable UIand APIs.
  + It integrates with **Gemini Flash 2.0 API** for AI-powered questiongeneration and feedback analysis.
  + Uses **React Speech Recognition** for speech-to-text functionality.
  + Backend is powered by **Drizzle ORM + PostgreSQL (Neon serverless DB)** for reliable data storage.

**Result**: Technically feasible with all tools being open-source, documented, and actively supported

#### Operational Feasibility

* + Endusers (students and job seekers) can easily access the platform through a web browser.
  + The system flow (Login → Start Interview → Answer → Feedback) is intuitive and requires no technical training.
  + Admin dashboard is not required for MVP, which reduces operational complexity.

**Result**: Operationally feasible with a simple onboarding flow.

#### Economic Feasibility

* + Development done using free/open-source tools.
  + Database hosting on **Neon** (serverless free tier).
  + Gemini Flash 2.0 API used via **Google AI Studio** (free tier fordevelopment).
  + No need for expensive licenses or infrastructure.

**Result**: Economically feasible for individual/academic deployment.

### Objectivesof Proposed System

The primary objectives of JARVIS are:

* + - To providean **automated mock interview experience** forcandidates.
    - To generate **role-specific questions** using AI.
    - To allow users to **speak answers** and transcribe them using speech-to-text.
    - To **evaluate answers** using AI and providereal-time feedback.
    - To **track user performance** across multiple sessions.
    - To offer a system that is scalable, user-friendly, and intelligent.

### Modules of Proposed System

The JARVIS platform consists of the following core modules:

#### Authentication Module

* 1. Handles user signup, login, and session management.

#### Dashboard Module

* 1. Shows existinginterview history.
  2. Allows users to create new interview sessions.

#### Interview Module

* 1. Generates questions based on the selected role.
  2. Captures answers via text or speech.
  3. Stores responses linked to session ID.

#### AI Feedback Module

* 1. Sends responses to Gemini Flash 2.0 model.
  2. Receives feedback and scores for each answer.
  3. Stores and displays results for the user.

#### Feedback Review Module

* 1. Presents feedback in a collapsible question-wise format.
  2. Includes detailed performance insights for each response.

#### Speech Input Integration

* 1. Uses react-speech-recognition to convert speech to text.
  2. Allows voice-based answering during interviews.

#### Database Interaction Module

* 1. Uses Drizzle ORM to manage schema and perform CRUD operations on PostgreSQL.

# Chapter 3: Analysis and Design

## Chapter 3: Analysis and Design

### System Requirements

#### Functional Requirements

Thesedefine the corefunctionalities that the JARVIS platform must support in order to deliver a seamless mock interview experience:

#### User Registration s Login:

The system must provide a secure authentication system, allowing users to register and log in. Users will have their unique profiles where interview history and progress are stored.

#### Interview Creation:

Users can create new interview sessions by selecting a predefined role (e.g., Software Developer, HR). Based on the selected role, the system generates role- specific questions dynamically.

#### Dynamic Question Generation:

The Gemini API is used to generate interview questions dynamically based on the user’s selected role. This feature eliminates the need for static question banks and allows for personalized mock interviews.

#### Answering Questions:

Users can answer questions via using speech-to-text (enabled through React Speech Recognition). This allows for a more interactive and realistic interview simulation.

#### Data Storage:

Each question, answer, and session metadata (user ID, timestamps, role) must be stored securely in the database for future access. This data can be used for performance tracking and historical review.

#### Feedback Generation:

After each interview session, the system should send user answers to the Gemini API for processing and receive feedback, which is then displayed to the user. The feedback will be question-wise, allowing users to review specific responses.

#### Performance Tracking:

The system will track key performance metrics such as accuracy, response time, and clarity of answers. Users should be ableto review their performance in previous sessions, making it easy to track progress over time.

#### Non-Functional Requirements

Theserequirements describe thegeneral properties of the system that ensure it meets quality standards:

#### Performance:

The system must load pages and respond to user actionsin a timely manner. This includes quick page loads, fast question rendering, and smooth transitions between different interview phases. APIs must respond in under 3 seconds.

#### Usability:

The platform shouldhave a clean and user-friendly interface. Tailwind CSS is used

to ensurea consistent, responsive design that works well across devices (mobile, tablet, and desktop). The design should allow users to easily navigate between interviews, feedback, and their performance history.

#### Scalability:

The system is designed to handle an increasing number of users and interviews. It is built for serverless deployment using Neon DB for cloud-based storage, making it easy to scale as the user base grows. The backend API is designed for high scalability and availability.

#### Maintainability:

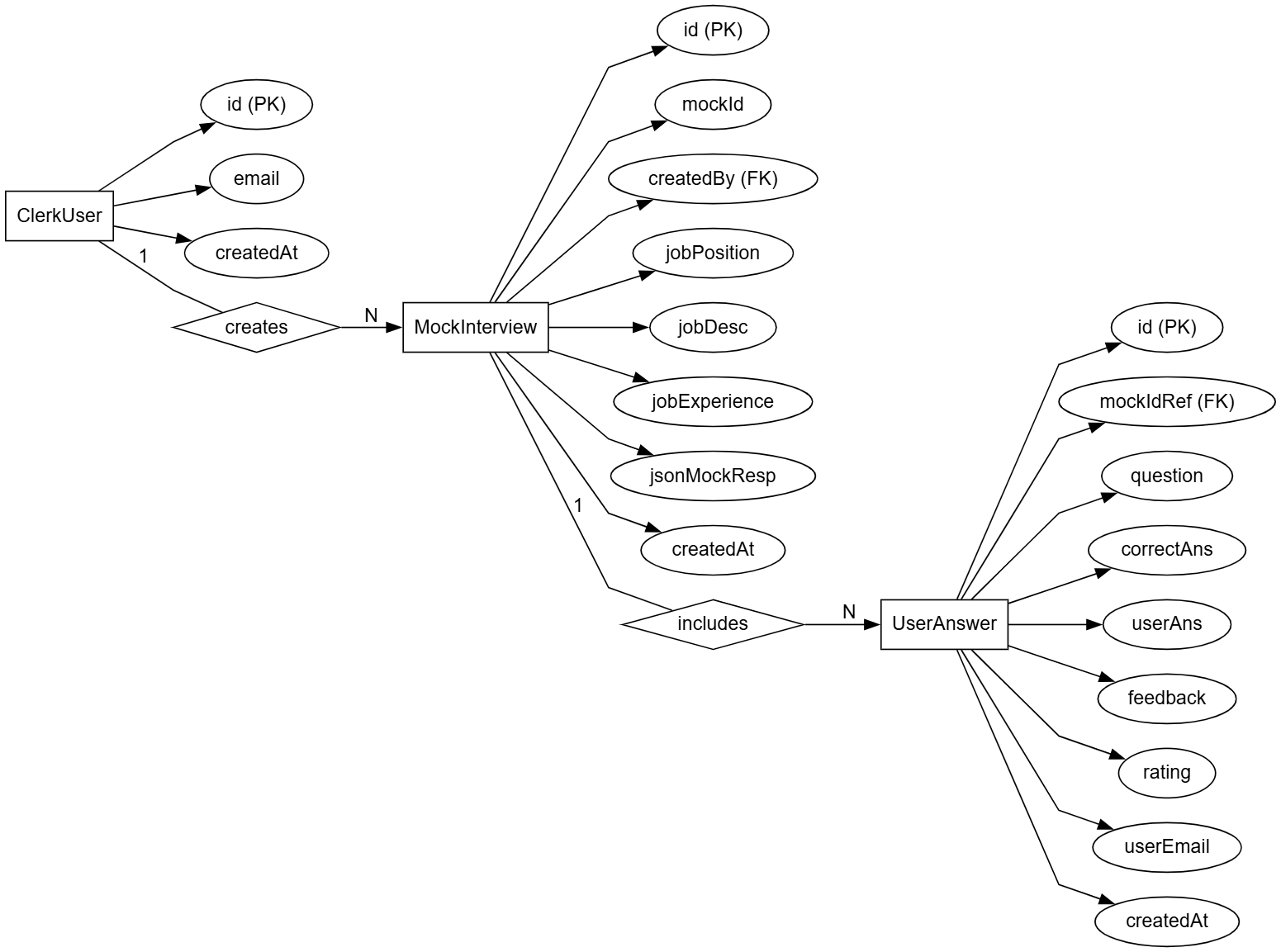
The codebase follows best practices for modular development. React components are reusable, and the Drizzle ORM provides a streamlined database interaction.

This ensures the system can be easily maintained and enhanced with minimal effort as new features are added.

#### Accessibility:

The system is browser-compatible, ensuring it works well across all major web browsers (Chrome, Firefox, Safari, Edge). The platform is also designed with accessibility in mind, supporting features like speech-to-text for users with disabilities. A mobile-friendly interface ensures it can be accessed from various devices, providing a seamless experience for all users

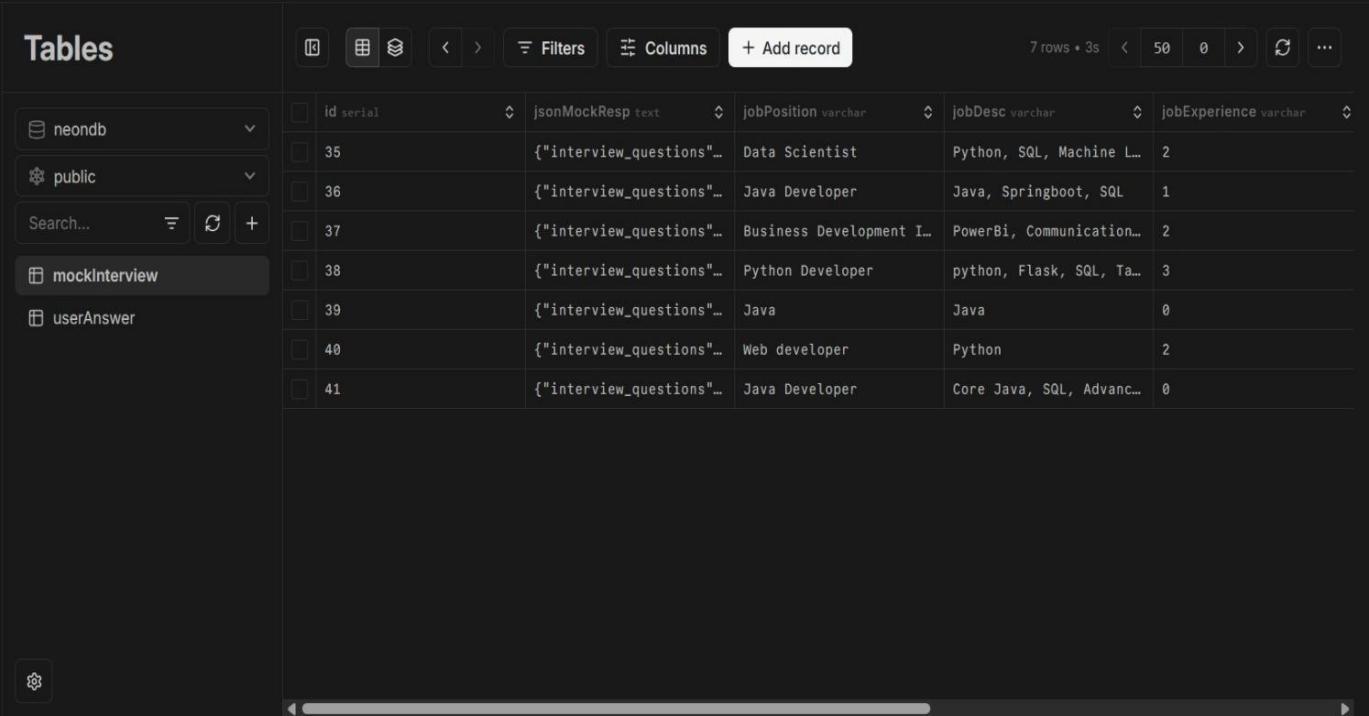
* 1. **Entity Relationship Diagram (ERD)**

****

* 1. **Table Structure**

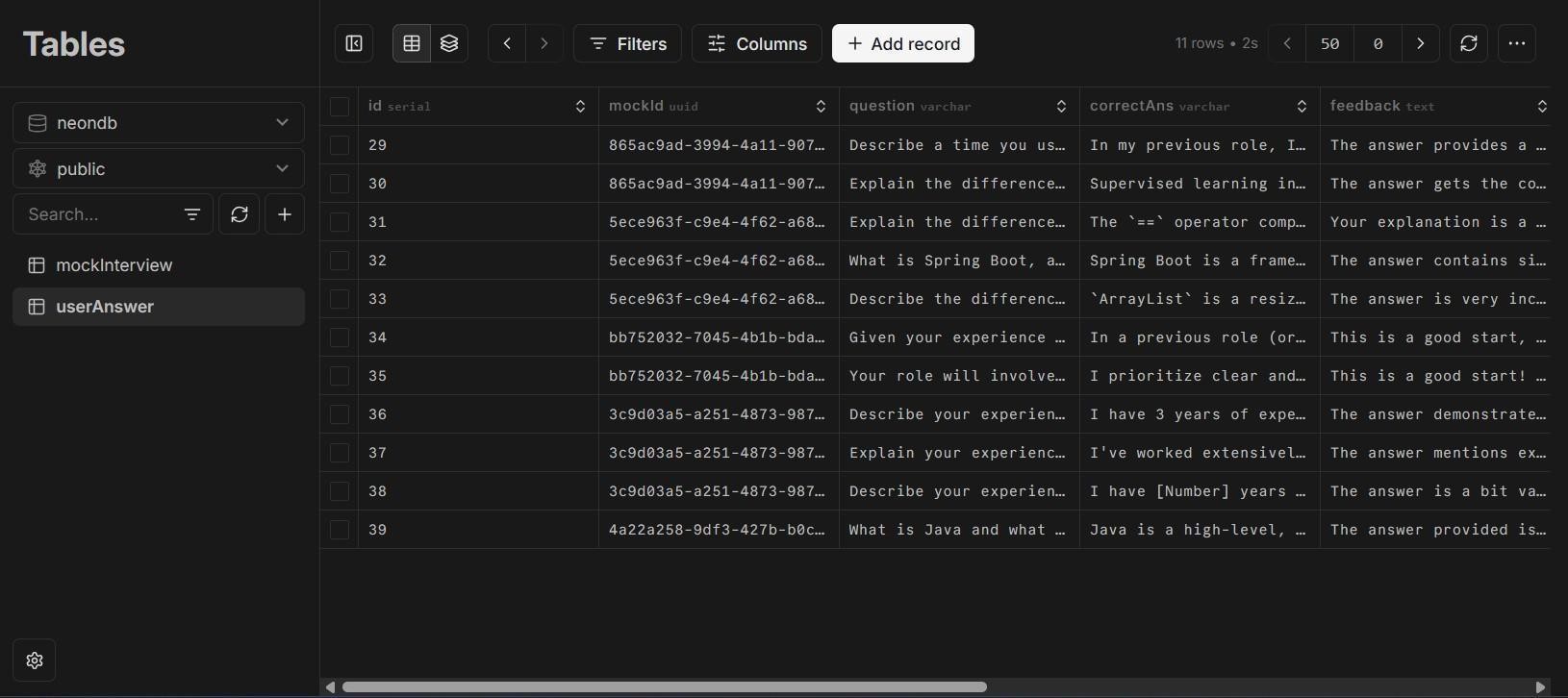
**Table 1: MockInterview**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table: mockInterview** | **Type** | **Key** | **Description** |
| id | SERIAL | PK | Internal surrogateprimarykey |
| mockId | UUID |  | Public session identifier (external key) |
| createdBy | VARCHAR | FK | Clerk user’s email (references user) |
| jobPosition | VARCHAR |  | Role/position for the mockinterview |
| jobDesc | VARCHAR |  | Shortjob description / tech stack |
| jobExperience | VARCHAR |  | Yearsof experiencerequired |
| jsonMockResp | TEXT |  | JSONblobof AI-generated QCA |
| createdAt | VARCHAR |  | Timestamp(YYYY-MM-DDHH:MM:SS) |

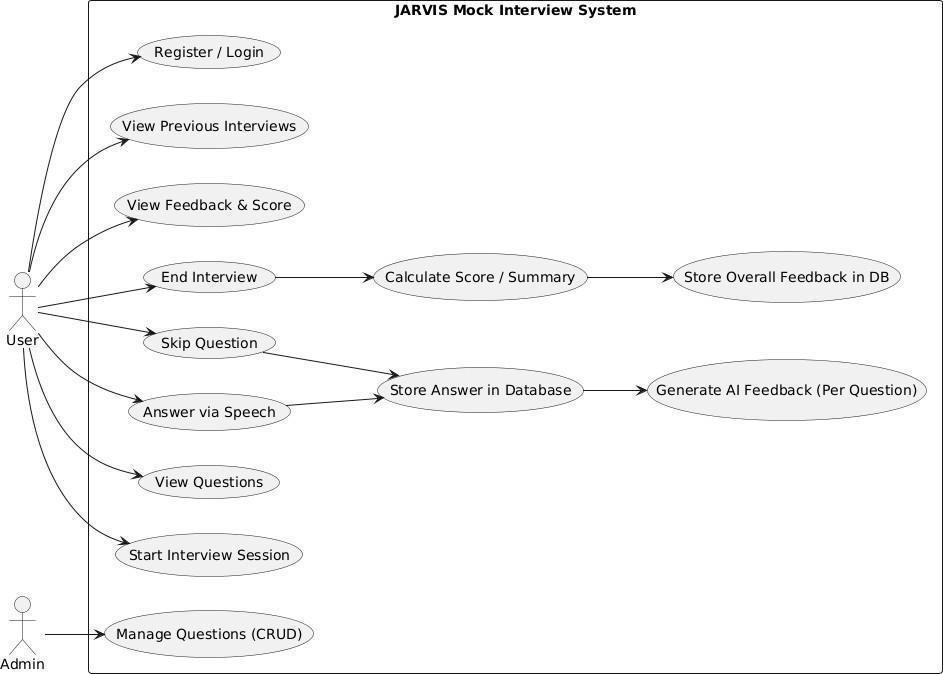
****

**Table 2: UserAnswer**

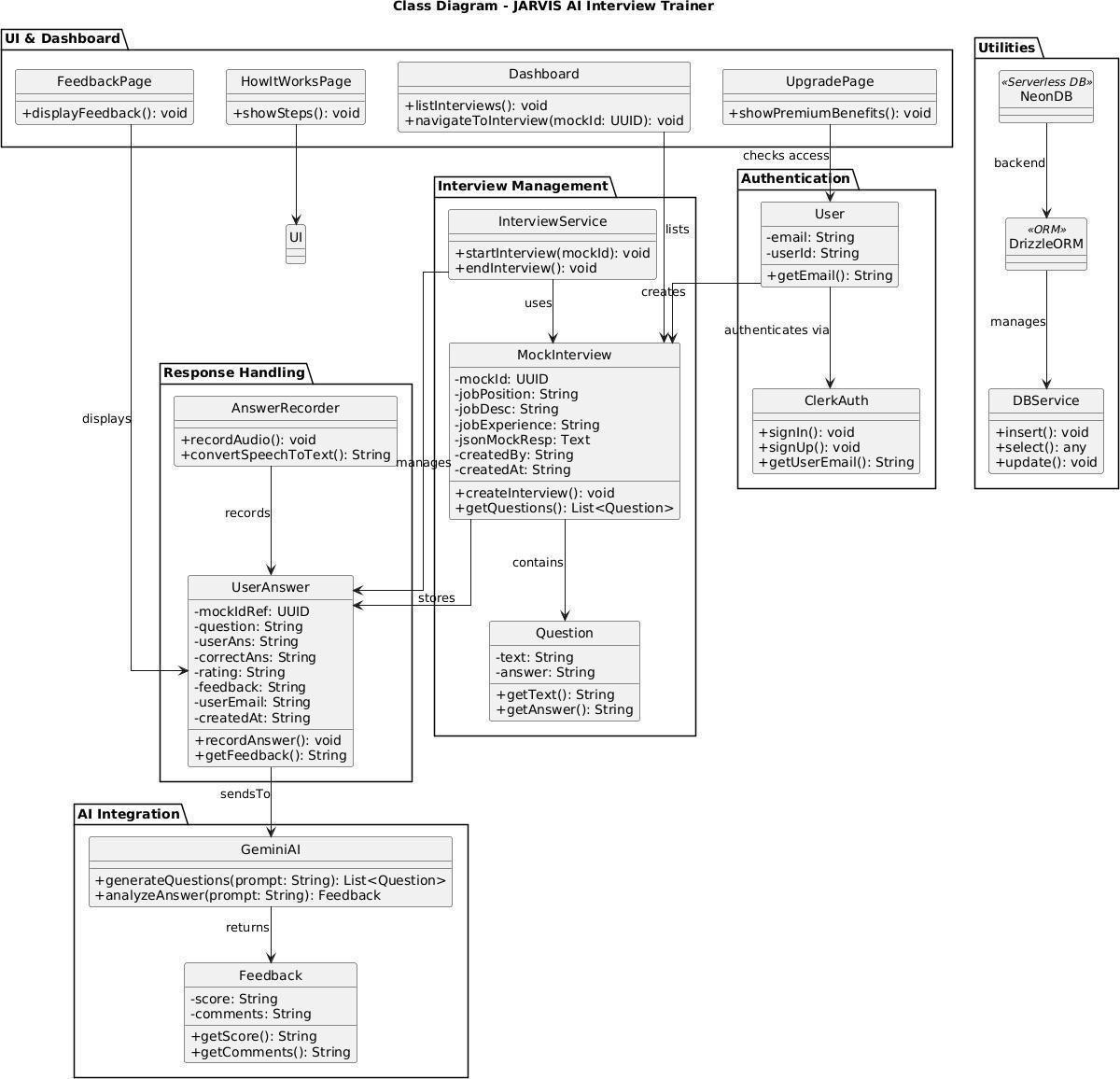
|  |  |  |  |
| --- | --- | --- | --- |
| **Table:**  **userAnswer** | **Type** | **Key** | **Description** |
| id | SERIAL | PK | Internalsurrogateprimarykey |
| mockIdRef | UUID | FK | References mockInterview.mockId |
| question | VARCHAR |  | Interview questiontext |
| correctAns | VARCHAR |  | AI-provided correct answer |
| userAns | TEXT |  | User’s response (speech-to-text or text) |
| feedback | TEXT |  | AI-generatedfeedback |
| rating | VARCHAR |  | AI-assignedrating(e.g.“8/10”) |
| userEmail | VARCHAR |  | Clerk user’s email who answered |
| createdAt | VARCHAR |  | Timestampwhen answer was recorded |

****

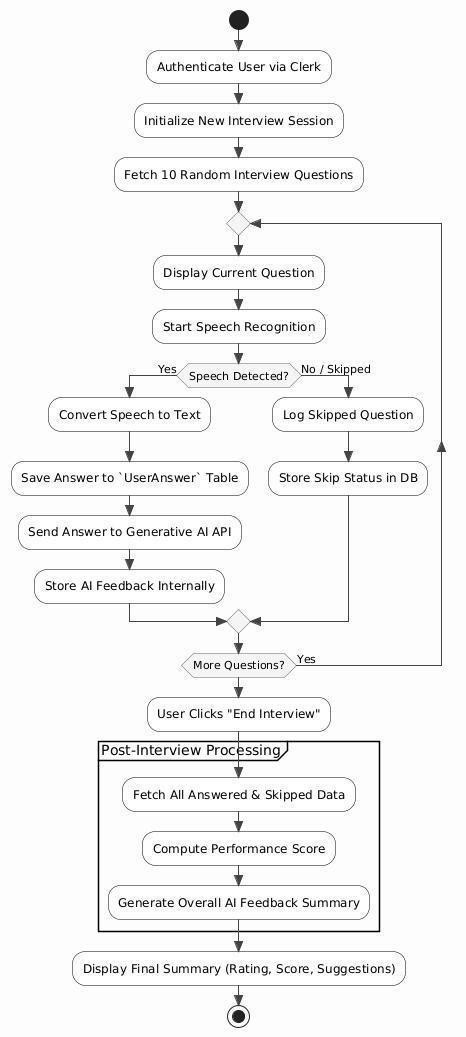
* 1. **Use Case Diagram**

****

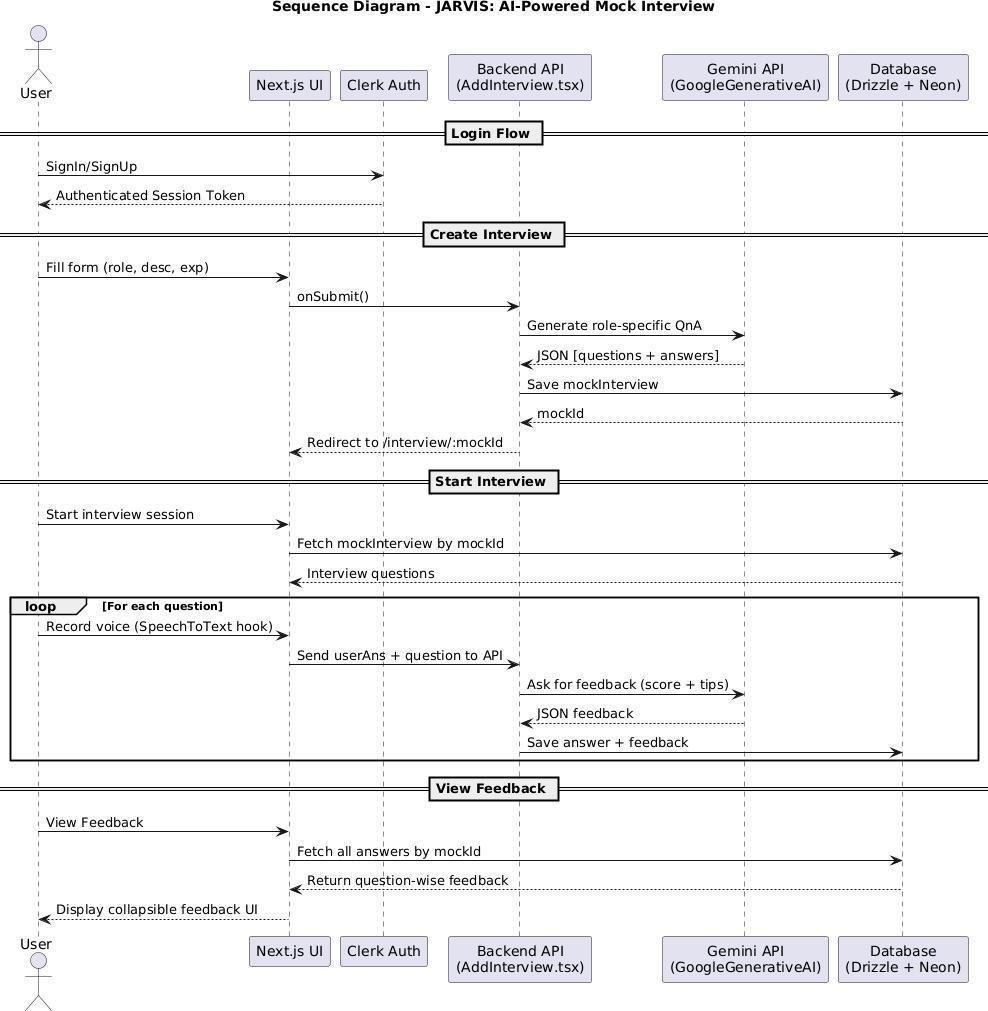
* 1. **Class Diagram**

****

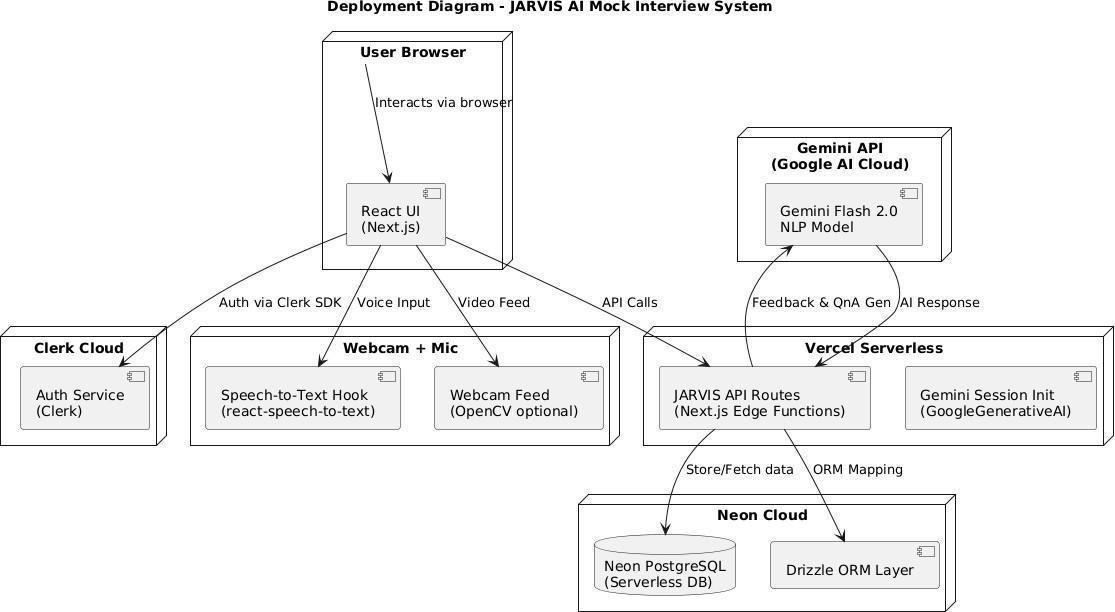
* 1. **Activity Diagram**

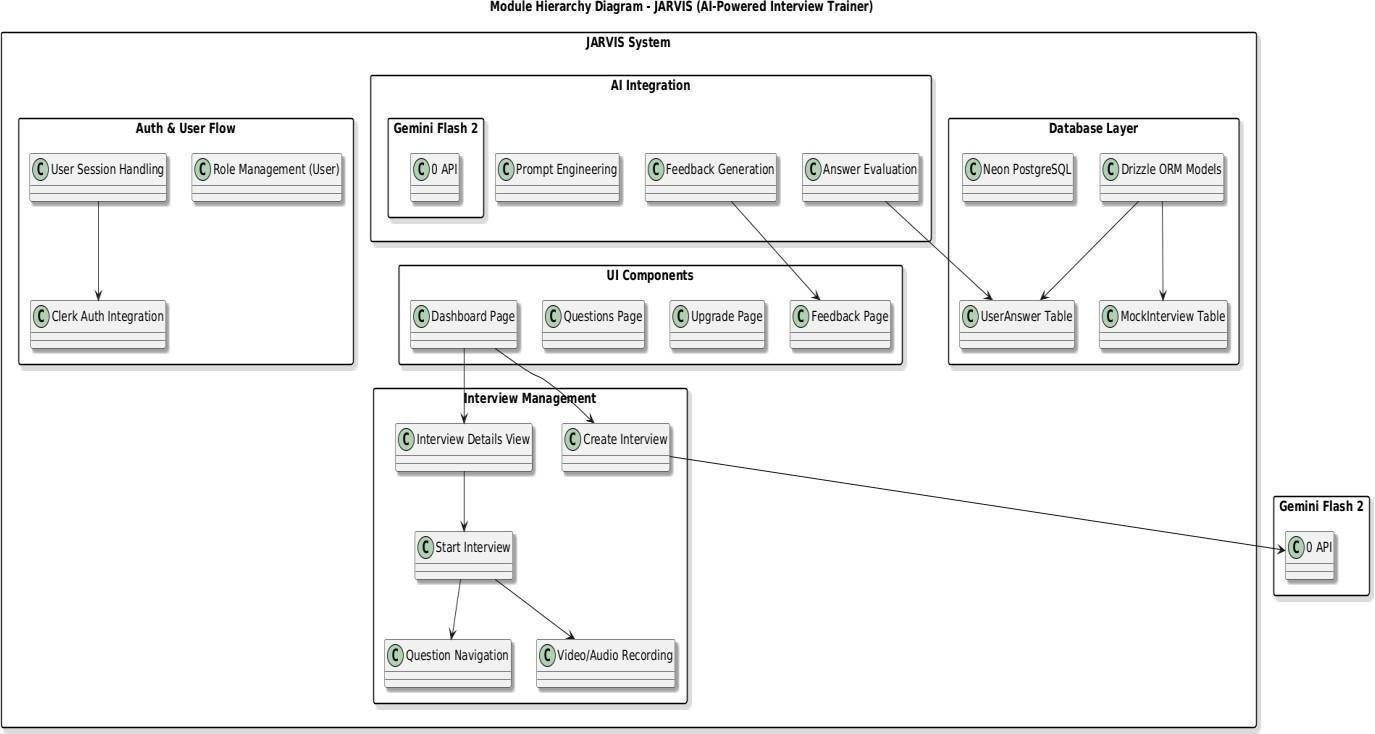
****

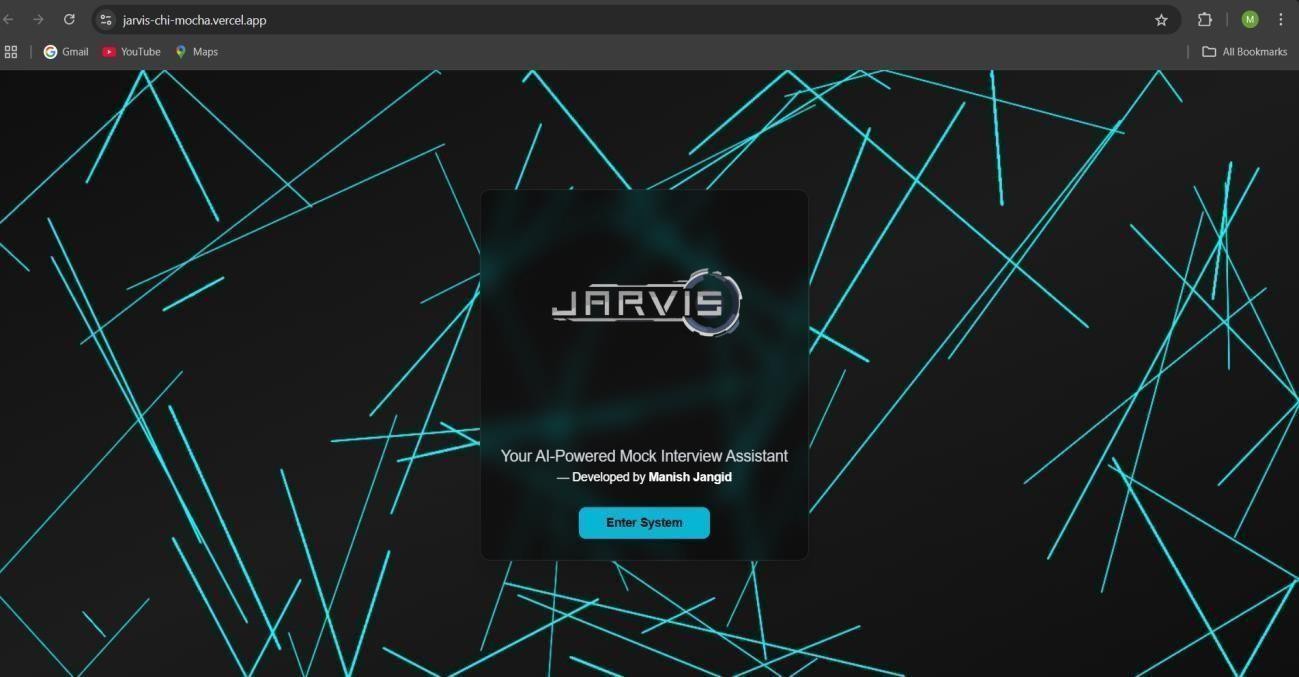
* 1. **Sequence Diagram**

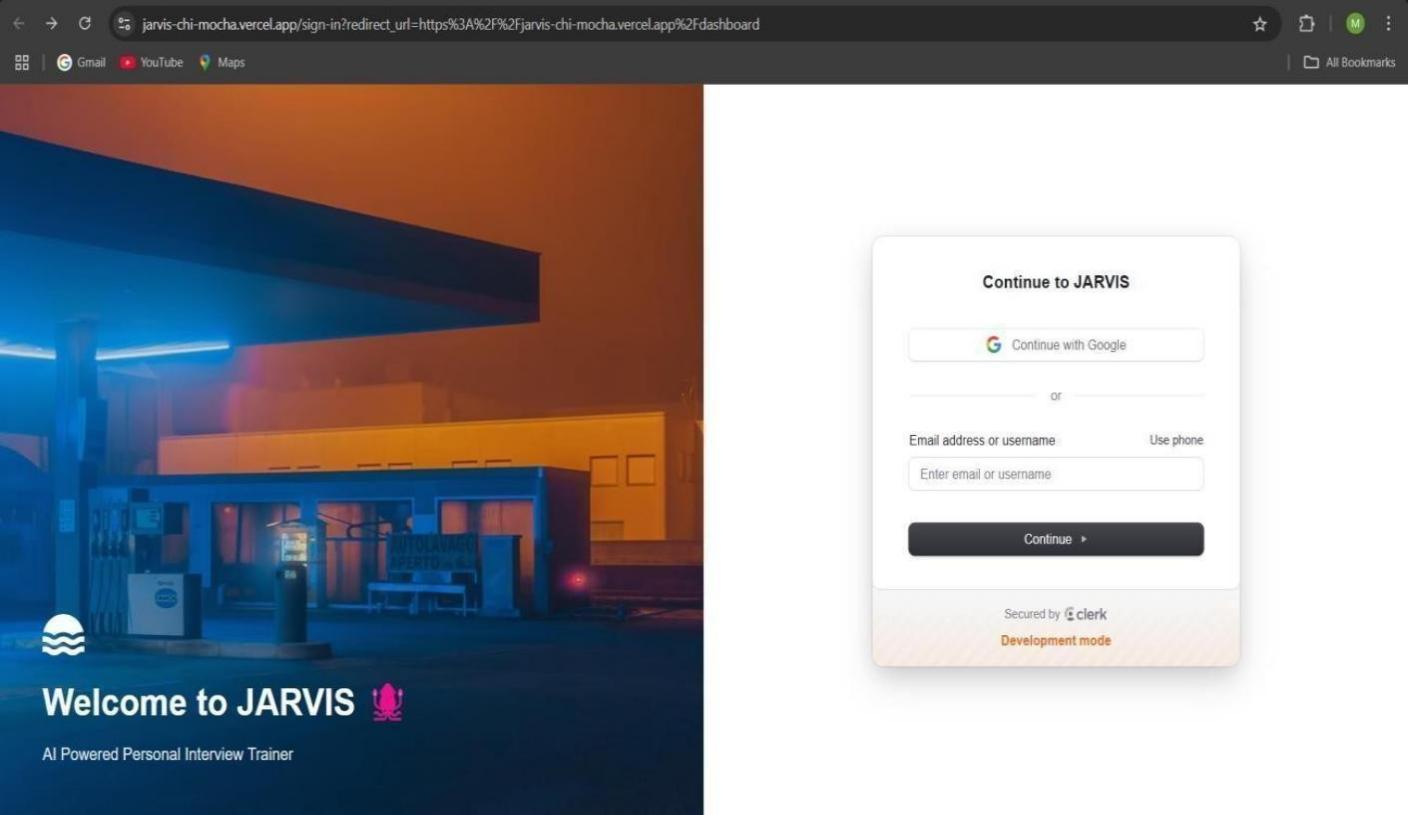
****

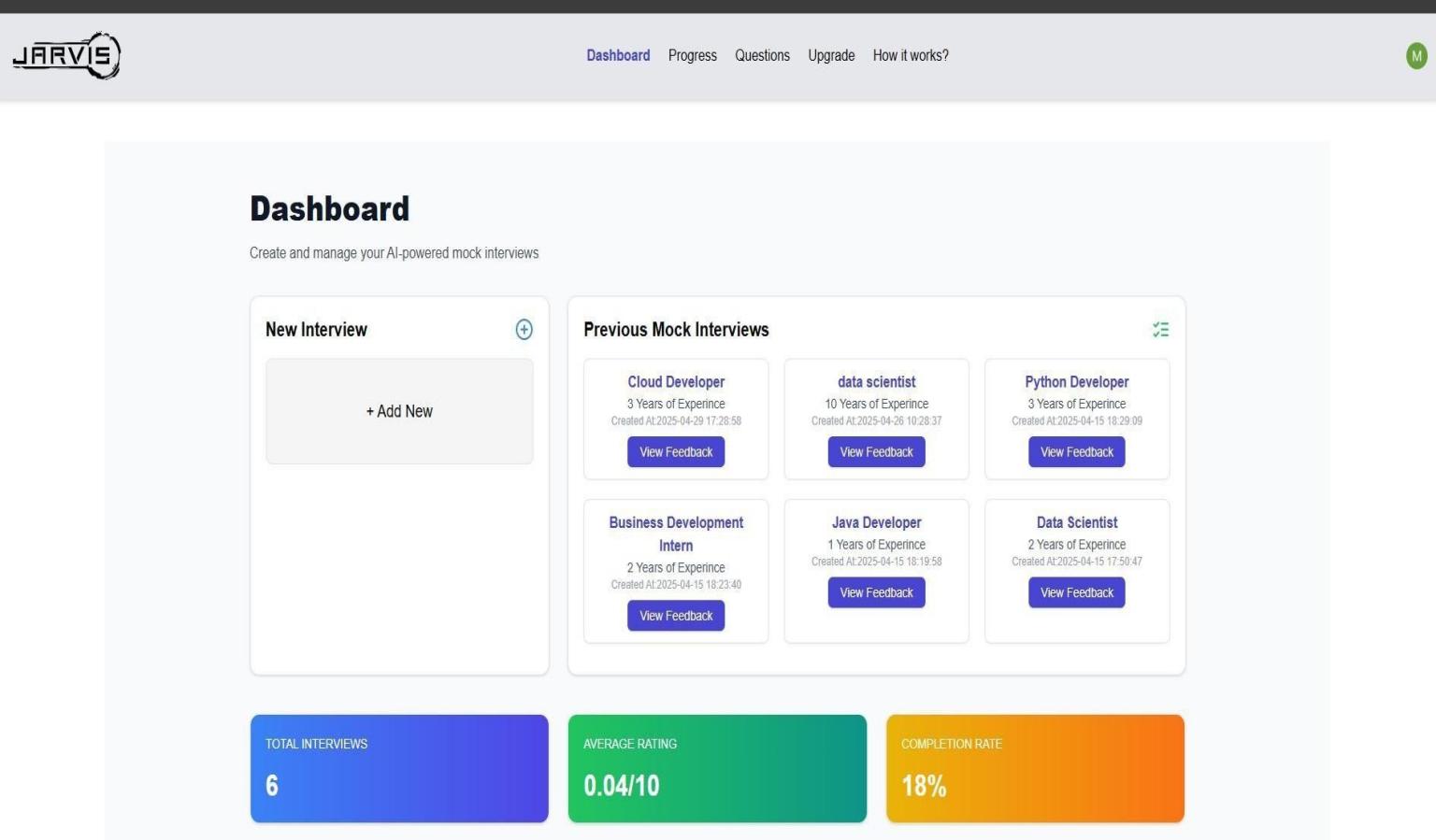
* 1. **Deployment Diagram**

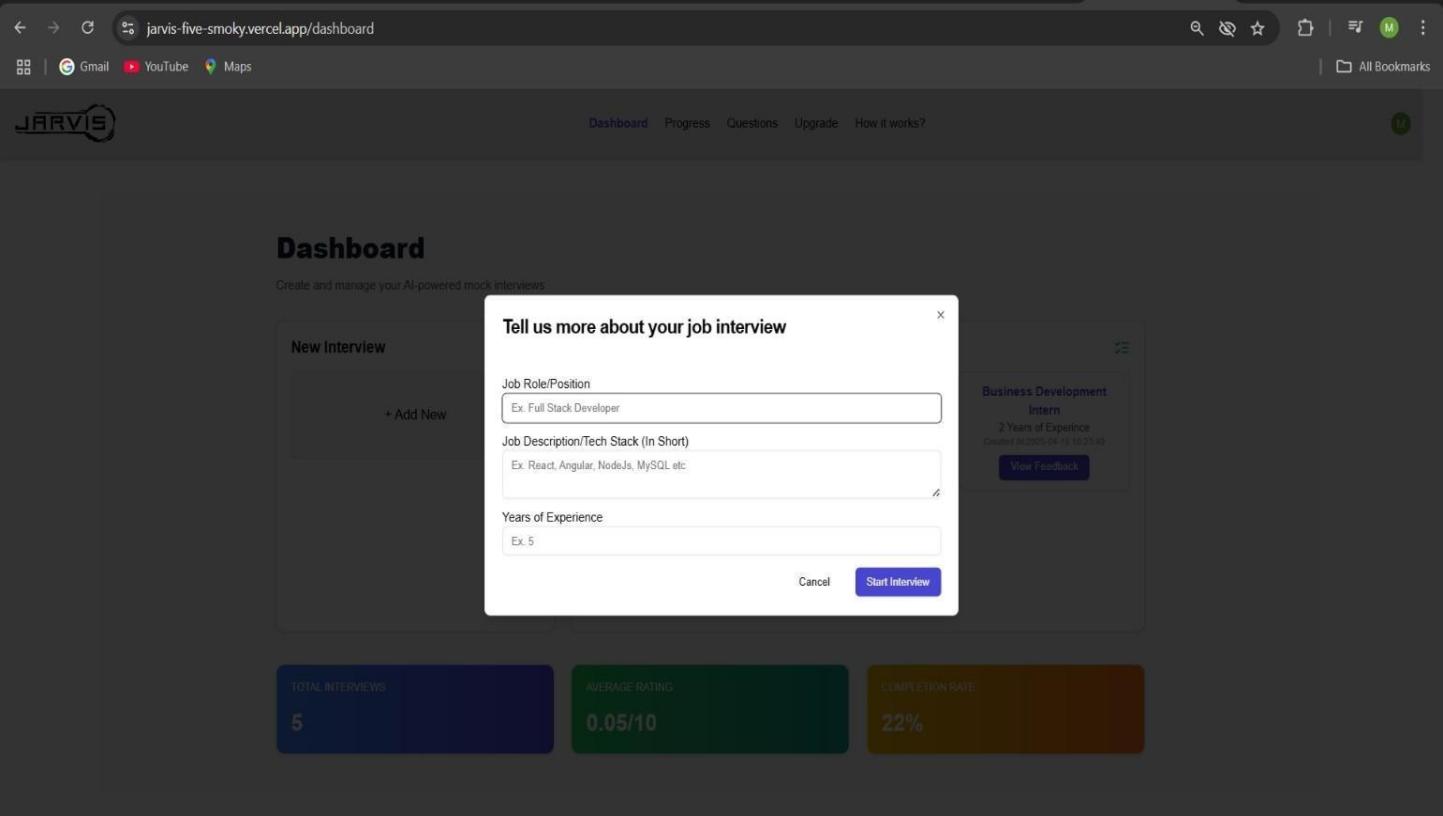
****

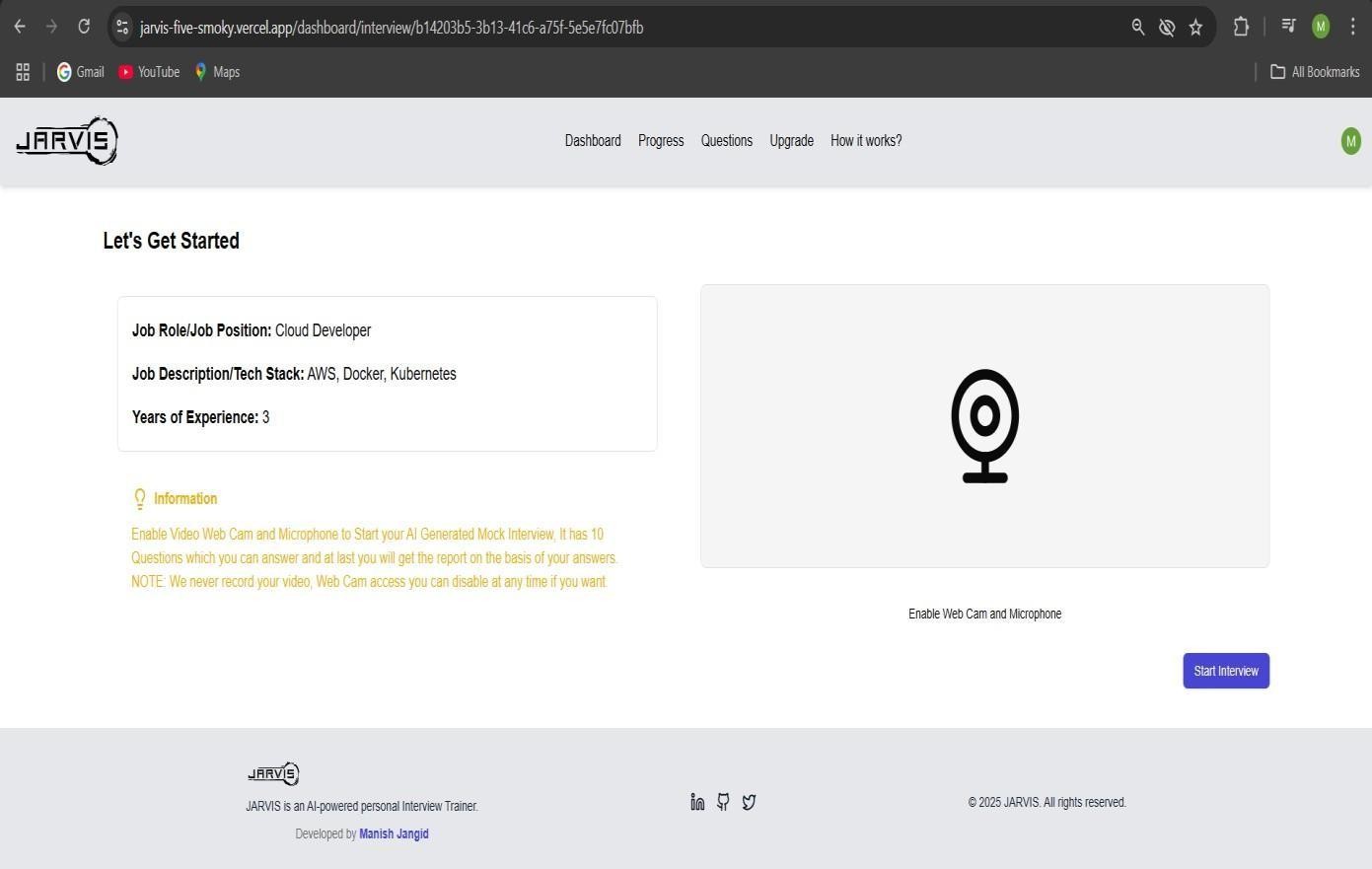
* 1. **ModuleHierarchy Diagram**
  2. **SampleInput and Output Screens**

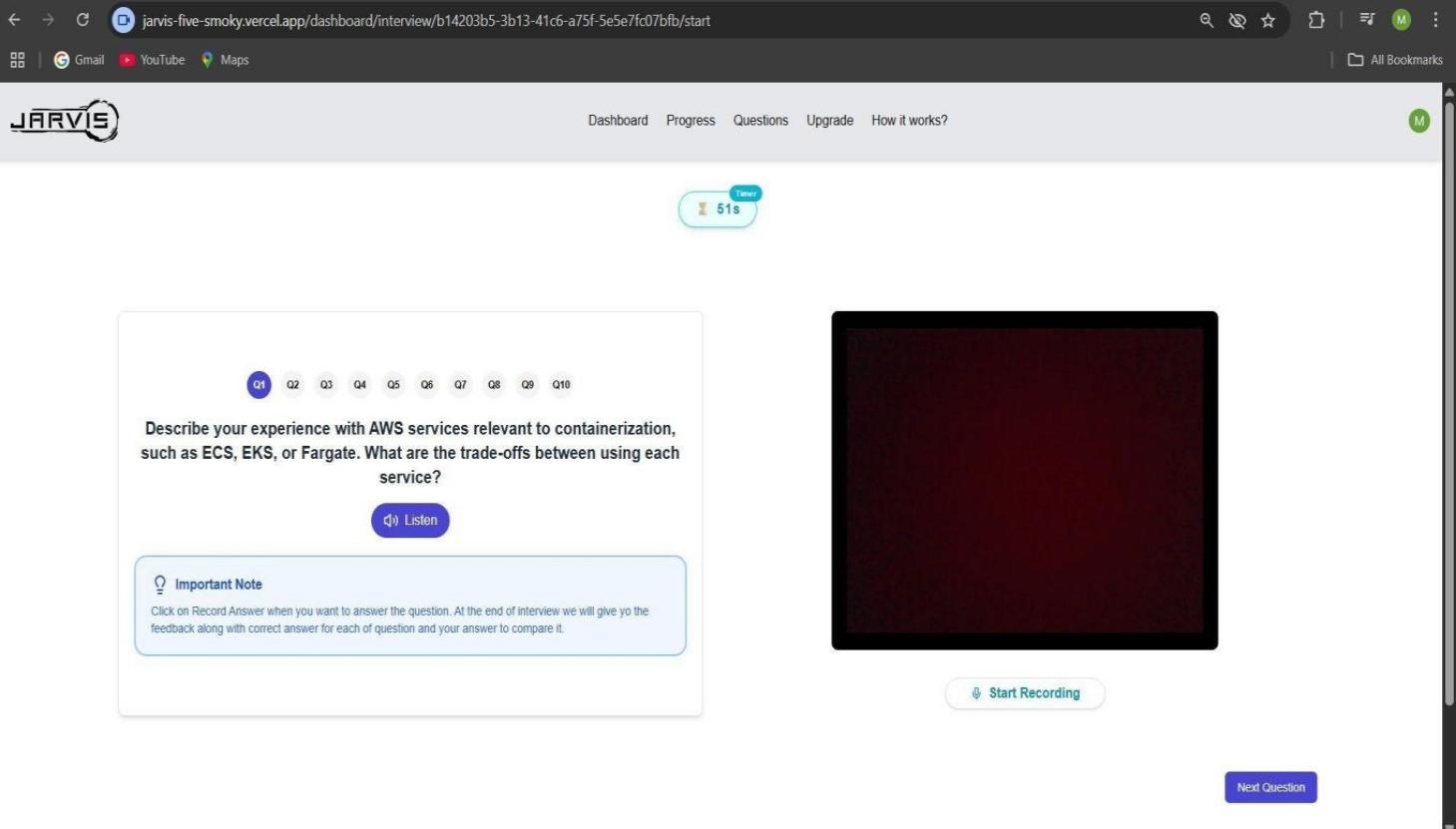
****

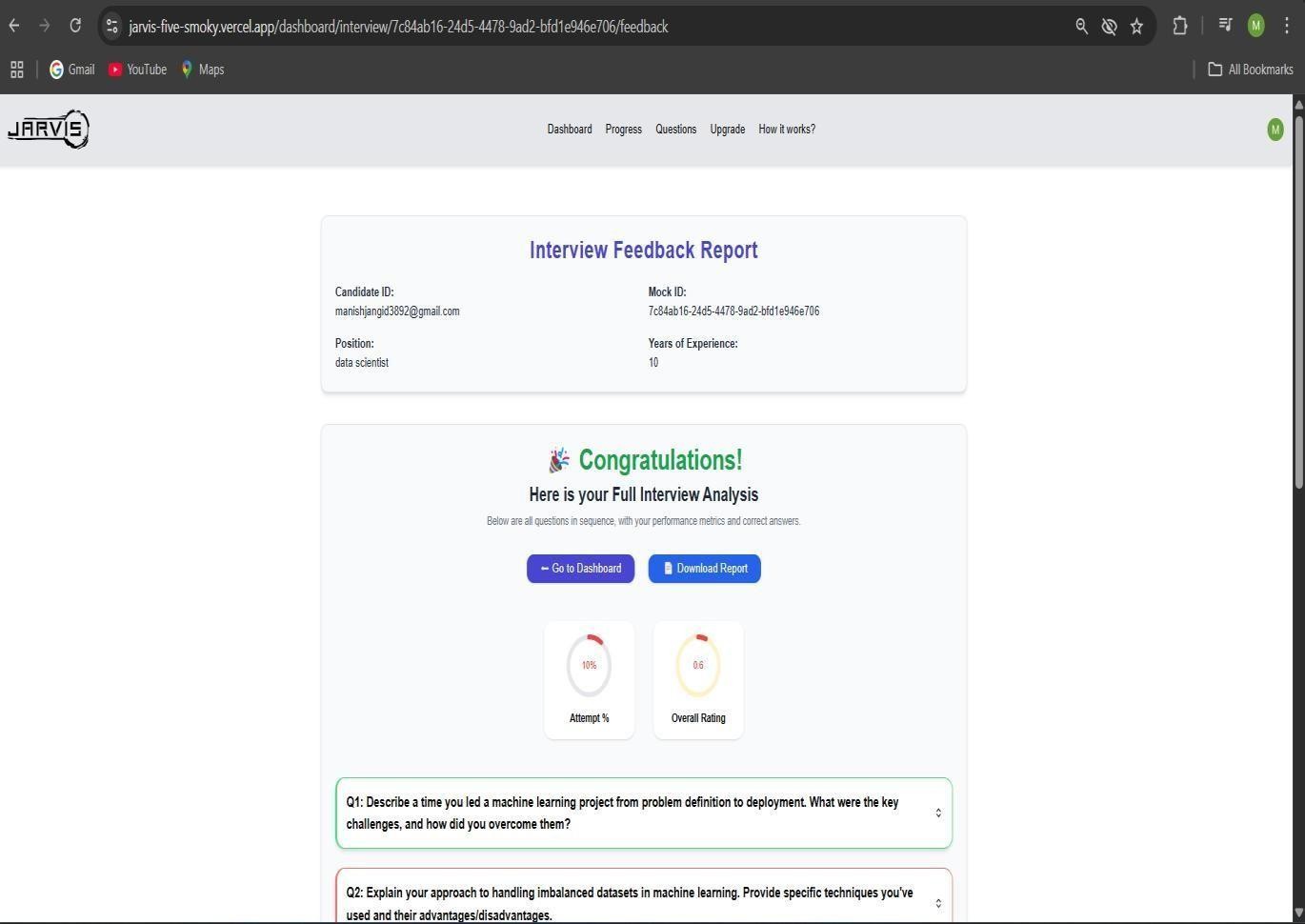
****

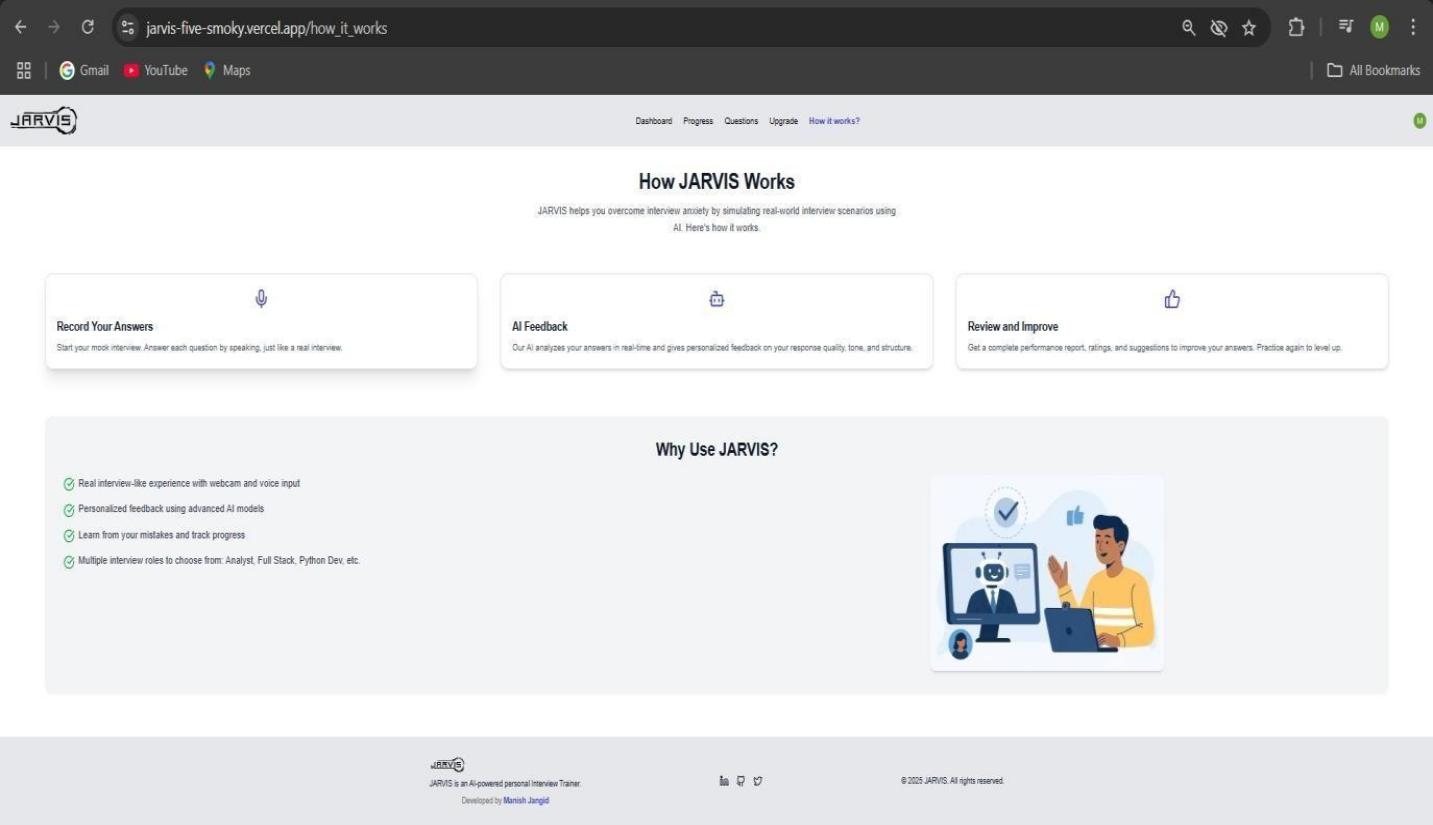


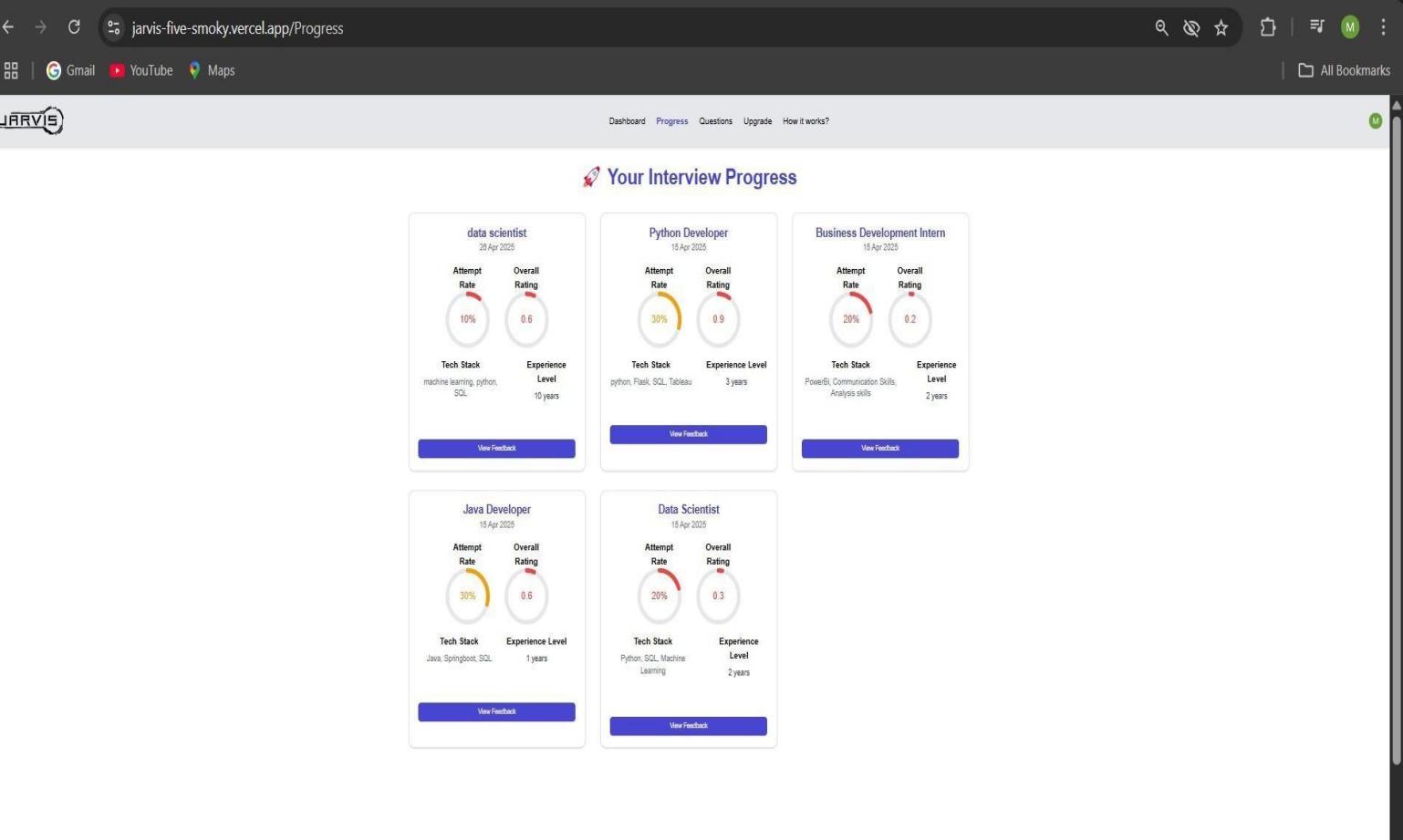
****

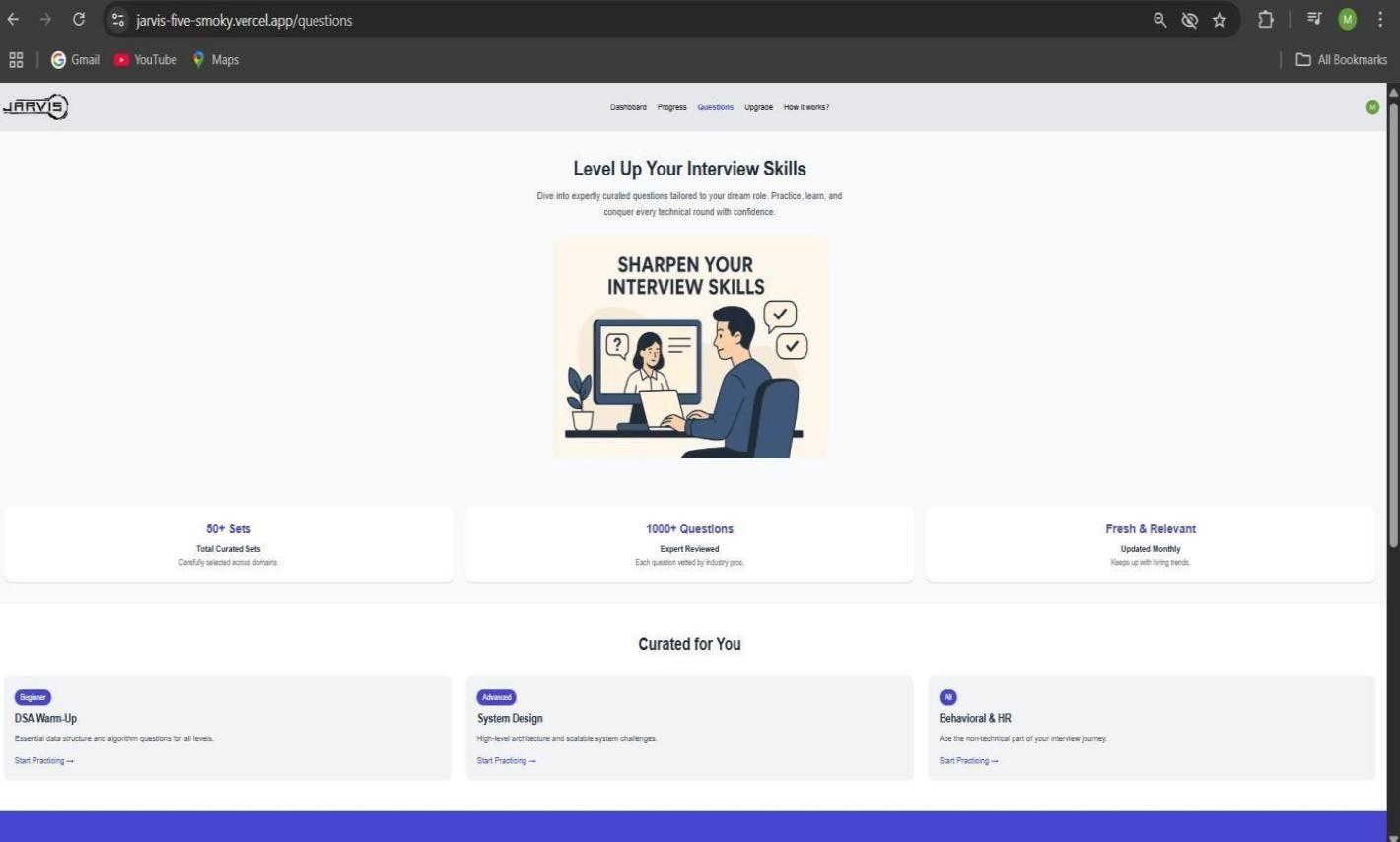


****



****



****

# Chapter 4: Coding

## Chapter4:Coding

### Code Snippets

This section provides an overview of the core code implementations used to bring the JARVIS mock interview system to life. The code is written using modern technologies such as React.js, Next.js, Drizzle ORM, Neon DB, Gemini AI API, and Clerk for authentication.

#### Database Configuration using Drizzle ORM

// db.js

import{ neon} from '@neondatabase/serverless'; import

{ drizzle } from 'drizzle-orm/neon-http'; import \* as schema from './schema';

constsql=neon(process.env.NEXT\_PUBLIC\_DRIZZLE\_DB\_URL); export const db = drizzle(sql,{schema} );

#### User Answer and Mock Interview Table Schema

// schema.js

import { pgTable, serial, text, varchar, uuid, timestamp } from 'drizzle-orm/pg-core'; export const MockInterview = pgTable('mockInterview', {

id: serial('id').primaryKey(),

mockId: uuid('mockId').defaultRandom(), jsonMockResp: text('jsonMockResp').notNull(), jobPosition: varchar('jobPosition').notNull(), jobDesc: varchar('jobDesc').notNull(), jobExperience: varchar('jobExperience').notNull(), createdBy: varchar('createdBy').notNull(), createdAt:varchar('createdAt')

});

export const UserAnswer=pgTable('userAnswer',{ id:serial('id').primaryKey(), mockIdRef:uuid('mockId').notNull(), question:varchar('question').notNull(), correctAns:varchar('correctAns'), userAns:text('UserAns'), feedback:text('feedback'), rating:varchar('rating'), userEmail:varchar('userEmail'), createdAt:varchar('createdAt')

});

#### Gemini AI Integration

// GeminiAIModel.js

import{ GoogleGenerativeAI } from"@google/generative-ai"; export function initializeChatSession() {

constapiKey=process.env.NEXT\_PUBLIC\_GEMINI\_API\_KEY; if (!apiKey) {

console.error(" APIKey is missing. MakesureNEXT\_PUBLIC\_GEMINI\_API\_KEY is set in .env.local");

return null;}

constgenAI=newGoogleGenerativeAI(apiKey);

constmodel= genAI.getGenerativeModel({ model:"gemini-2.0-flash"});

returnmodel.startChat({ generationConfig: { temperature: 1,

topP: 0.95,

topK: 40,

maxOutputTokens: 8192, responseMimeType: "text/plain",

},

});

}

#### Clerk Authentication in RootLayout

// layout.js

import { Geist, Geist\_Mono } from "next/font/google"; import { ClerkProvider } from "@clerk/nextjs";

import "./globals.css";

import dynamic from 'next/dynamic';

import { Toaster } from "@/components/ui/sonner";

const geistSans = Geist({ variable: "-

-font-geist-sans", subsets: ["latin"],

});

constgeistMono= Geist\_Mono({ variable: "--font- geist-mono", subsets: ["latin"],

});

exportconstmetadata= { title:

"Create Next App",

description: "Generated by createnext app",

};

export default function RootLayout({ children }) { return (

<ClerkProvider>

<html lang="en">

<body

className={`${geistSans.variable} ${geistMono.variable} antialiased`}

>

<Toaster />

{children}

</body>

</html>

</ClerkProvider>

)}

#### Start Interview: AI Question Generation s JSON Parsing

//Start Interview.jsx "use client";

import React, { use, useEffect, useState } from 'react'

import { useParams } from "next/navigation"; // ✅ Import useParams() instead of props import { MockInterview } from "@/utils/schema";

import { db } from "@/utils/db"; import { eq } from "drizzle-orm";

import RecordAnswerSection from './\_components/RecordAnswerSection'; import QuestionsSection from './\_components/QuestionsSection';

import { Button } from '@/components/ui/button'; import Link from "next/link";

function StartInterview() {

const params = useParams(); // ✅ Get params from Next.js const interviewId = params?.interviewId || "";

const [interviewData, setInterviewData] = useState();

const [mockInterviewQuestion,setMockInterviewQuestion] = useState([]); const [activeQuestionIndex,setActiveQuestionIndex] = useState(0);

const [timeLeft, setTimeLeft] = useState(60); // 60 seconds for each question const [questionsAttempted, setQuestionsAttempted] = useState(0);

useEffect(() => {

if (!interviewId || typeof interviewId !== "string") { console.warn("Invalid interviewId:", interviewId); return;

}

console.log("Fetching interview for ID:", interviewId); GetInterviewData(interviewId);

}, [interviewId]);

useEffect(() => {

setTimeLeft(60); // Reset timer on new question

}, [activeQuestionIndex]);

useEffect(() => {

if (timeLeft === 0) {

if (activeQuestionIndex < mockInterviewQuestion.length - 1) { setActiveQuestionIndex((prev) => prev + 1); // Move to next question

}

}

const timer = setTimeout(() => {

}, 1000);

return () => clearTimeout(timer);

}, [timeLeft]);

const GetInterviewData = async (interviewId) => { try {

const result = await db.select().from(MockInterview).where(eq(MockInterview.mockId, interviewId));

if (result.length === 0) {

console.warn(" No interview found for this ID."); return;

}

console.log("✅ Fetched Interview Details:", result[0]); let jsonMockResp;

try {

jsonMockResp = result[0]?.jsonMockResp ? JSON.parse(result[0].jsonMockResp) : null;

} catch (error) {

console.error("❌ Error parsing JSON:", error); return;

}

if (!jsonMockResp || Object.keys(jsonMockResp).length === 0) { console.error("❌ jsonMockResp is empty or invalid:", jsonMockResp); return;

}

if (!Array.isArray(jsonMockResp.interview\_questions)) {

console.error("❌ interview\_questions is missing or not an array:", jsonMockResp); return;

}

console.log("✅ Valid interview questions found:", jsonMockResp.interview\_questions); setMockInterviewQuestion(jsonMockResp.interview\_questions); setInterviewData(result[0]);

} catch (error) {

console.error("❌ Error fetching interview:", error);

{/\* Timer Section \*/}

<div className="flex items-center justify-center my-4">

<div className={`relative px-5 py-2 rounded-full shadow-md border ${timeLeft <= 10 ? "border-red- 500 bg-red-50 animate-pulse" : "border-cyan-400 bg-cyan-50"}`}>

<h2 className={`text-sm md:text-base font-semibold tracking-widest ${timeLeft <= 10 ? "text-red- 600" : "text-cyan-600"}`}>

⏳ {timeLeft < 10 ? `0${timeLeft}` : timeLeft}s

</h2>

<div className={`absolute -top-2 -right-2 text-[10px] px-2 py-0.5 rounded-full font-bold shadow

${timeLeft <= 10 ? "bg-red-500 text-white" : "bg-cyan-500 text-white"}`}> Timer

</div>

</div>

</div>

{/\* Main Sections \*/}

<div className="grid grid-cols-1 md:grid-cols-2 gap-10">

<QuestionsSection mockInterviewQuestion={mockInterviewQuestion} activeQuestionIndex={activeQuestionIndex}

/>

<RecordAnswerSection mockInterviewQuestion={mockInterviewQuestion} activeQuestionIndex={activeQuestionIndex} interviewData={interviewData}

onAnswerRecorded={() => setQuestionsAttempted(prev => prev + 1)} // questionsAttempted={questionsAttempted} // optional if needed

/>

</div>

{/\* Navigation Buttons \*/}

<div className="flex justify-end gap-6 mt-10">

{activeQuestionIndex !== mockInterviewQuestion.length - 1 && (

<Button onClick={() => setActiveQuestionIndex(activeQuestionIndex + 1)}> Next Question

</Button>

)}

{activeQuestionIndex === mockInterviewQuestion?.length - 1 && interviewData?.mockId && (

<Button

disabled={questionsAttempted < 0} // Minimum 2 questions must be attempted onClick={() => {

if (questionsAttempted < 2) {

alert(`Please attempt at least 2 questions before ending the interview.`); return;

}

{/\* Use Link to wrap the Button when conditions are met \*/}

{questionsAttempted >= 2 ? (

<Link href={`/dashboard/interview/${interviewData.mockId}/feedback`}> End Interview

</Link>

) : (

"End Interview"

)}

</Button>

)}

</div>

</div>

);

}

export default StartInterview

#### Recording Answer with Speech-to-Text

//RecordAnswerSection.jsx "use client";

import React, { useEffect, useState } from "react";

import Image from "next/image"; import WebCam from "react-webcam";

import { Button } from "@/components/ui/button"; import useSpeechToText from "react-hook-speech-to-text"; import { Mic } from "lucide-react";

import { toast } from "sonner";

import { useUser } from "@clerk/nextjs"; import { db } from "@/utils/db";

import moment from "moment";

import { initializeChatSession } from "@/utils/GeminiAIModel";

import { UserAnswer } from "@/utils/schema"; // ✅ Importing Drizzle ORM schema import { MockInterview } from "@/utils/schema";

function RecordAnswerSection({ mockInterviewQuestion, activeQuestionIndex, interviewData, onAnswerRecorded }) {

const [userAnswer, setUserAnswer] = useState(""); const { user } = useUser();

const [loading, setLoading] = useState(false); const {

error, interimResult, isRecording, results, startSpeechToText, stopSpeechToText, setResults,

} = useSpeechToText({ continuous: true, useLegacyResults: false,

});

useEffect(() => {

if (results.length > 0) {

setUserAnswer(results.map((r) => r.transcript).join(" "));

}

}, [results]);

useEffect(() => {

if (!isRecording && userAnswer?.length > 10) { UpdateUserAnswer();

}

}, [userAnswer])

const StartStopRecording = async () => {

if (isRecording) { stopSpeechToText();

} else { startSpeechToText();

}

};

const UpdateUserAnswer = async () => { console.log(" User Answer:", userAnswer); setLoading(true);

const feedbackPrompt = `Question: ${mockInterviewQuestion[activeQuestionIndex]?.question}, User Answer: ${userAnswer},

Provide a JSON response with "rating"(out of 10) and "feedback" fields.`;

try {

const chatSession = initializeChatSession(); // ✅ Initialize AI Chat Session if (!chatSession) throw new Error("AI Chat session initialization failed");

const result = await chatSession.sendMessage(feedbackPrompt); let mockJsonResp = result.response.text();

console.log("Raw AI Response:", mockJsonResp);

// Clean the response by removing unwanted characters and ensuring proper JSON format mockJsonResp = mockJsonResp.replace(/```json|```/g, '').trim();

try {

// Attempt to parse the cleaned JSON

const JsonFeedbackResp = JSON.parse(mockJsonResp);

// ✅ Insert into database using Drizzle ORM await db.insert(UserAnswer).values({ mockIdRef: interviewData?.mockId,

question: mockInterviewQuestion[activeQuestionIndex]?.question, correctAns: mockInterviewQuestion[activeQuestionIndex]?.answer, userAns: userAnswer,

feedback: JsonFeedbackResp?.feedback, rating: JsonFeedbackResp?.rating,

userEmail: user?.primaryEmailAddress?.emailAddress, createdAt: moment().format("YYYY-MM-DD HH:mm:ss"),

setUserAnswer(""); setResults([]);

if (onAnswerRecorded) {

onAnswerRecorded(); // Increase attempted questions after success

}

} catch (jsonParseError) {

console.error("❌ Invalid JSON format:", jsonParseError); toast.error("Failed to parse AI response. Please try again.");

}

} catch (error) {

console.error("❌ Error Updating Answer:", error); toast.error("Something went wrong!");

}

setLoading(false);

};

return (

<div className="flex items-center justify-center flex-col">

<div className="flex flex-col mt-20 justify-center items-center bg-black rounded-lg p-5">

<Image src={"/WebCam.png"} width={300} height={300} className="absolute" alt="WebCam" />

<WebCam mirrored={true} style={{ height: 350,

width: "100%",

zIndex: 10,

}}

/>

</div>

{/\* Record Button \*/}

<Button disabled={loading} variant="outline"

className="my-8 px-8 py-3 text-base font-semibold rounded-full shadow hover:shadow-md transition-all"

onClick={StartStopRecording}

>

{isRecording ? (

<span className="text-red-600 flex items-center gap-2">

<Mic className="h-5 w-5" /> Stop Recording

</span>

) : (

<span className="flex items-center gap-2 text-cyan-600">

<Mic className="h-5 w-5" /> Start Recording

</span>

)}

</Button>

</div>

);

}

export default RecordAnswerSection;

#### Feedback Design s Display

//Feedback.jsx "use client";

import { useState, useEffect, useRef } from 'react'; import { db } from "@/utils/db";

import { UserAnswer, MockInterview } from "@/utils/schema"; import { eq } from "drizzle-orm";

import { useParams, useRouter } from "next/navigation"; import { jsPDF } from "jspdf";

import "jspdf-autotable"; // Import the autoTable plugin for tables in PDF import {

Collapsible, CollapsibleContent, CollapsibleTrigger,

} from "@/components/ui/collapsible";

import { ChevronsUpDown } from "lucide-react"; import { Button } from "@/components/ui/button";

import { CircularProgressbar, buildStyles } from "react-circular-progressbar"; import "react-circular-progressbar/dist/styles.css";

import clsx from "clsx";

import html2canvas from 'html2canvas'; import Skeleton from 'react-loading-skeleton';

import 'react-loading-skeleton/dist/skeleton.css'; import { ToastContainer, toast } from 'react-toastify'; import 'react-toastify/dist/ReactToastify.css';

import html2pdf from 'html2pdf.js'; // Add this import

// Helper function to pick red/amber/green color const getColorForPercentage = (pct) => {

if (pct < 30) return "#ef4444"; // red-500 if (pct < 70) return "#f59e0b"; // amber-500 return "#10b981"; // green-500

};

function Feedback() {

const { interviewId } = useParams(); const router = useRouter();

const [mergedList, setMergedList] = useState([]);

const [totalQuestions, setTotalQuestions] = useState(0);

const [attemptedQuestions, setAttemptedQuestions] = useState(0); const [averageRating, setAverageRating] = useState(0);

const [userDetails, setUserDetails] = useState({});

const [loading, setLoading] = useState(true); // Track loading state const feedbackRef = useRef(null);

const [isGenerating, setIsGenerating] = useState(false);

const [openAll, setOpenAll] = useState(false); // control expand state for all collapsibles

useEffect(() => {

if (interviewId) fetchFeedbackData();

}, [interviewId]);

const fetchFeedbackData = async () => {

setLoading(true); // Set loading to true when data is being fetched

// 1) Fetch attempted answers from UserAnswer table const userAnswers = await db

.select()

.from(UserAnswer)

.where(eq(UserAnswer.mockIdRef, interviewId))

.orderBy(UserAnswer.id);

// 2) Fetch interview questions and correct answers from MockInterview table const interview = await db

.select()

.from(MockInterview)

.where(eq(MockInterview.mockId, interviewId));

const originalQuestions = JSON.parse(interview[0]?.jsonMockResp)?.interview\_questions || [];

// 3) Fetch user details const user = await db

.select()

.from(UserAnswer)

.where(eq(UserAnswer.mockIdRef, interviewId))

.limit(1); // Assuming user info can be fetched from the UserAnswer table

// 4) Compute summary metrics

const total = originalQuestions.length;

const attempted = userAnswers.filter((a) => a.userAns?.length > 0).length; const avgRating =

userAnswers.length > 0

? (

userAnswers.reduce((sum, a) => sum + parseFloat(a.rating || "0"), 0) /

10

).toFixed(1)

: 0;

setTotalQuestions(total); setAttemptedQuestions(attempted); setAverageRating(avgRating);

// Set user details (adjust field names as necessary) setUserDetails({

jobDesc: interview[0]?.jobDesc || "N/A", jobTitle: interview[0]?.jobPosition || "N/A", experience: interview[0]?.jobExperience || "N/A", mockId: interviewId,

createdBy: interview[0]?.createdBy || "N/A",

});

// 5) Merge questions with user answers and correct answers from jsonMockResp const merged = originalQuestions.map((q, idx) => {

const attempt = userAnswers.find((a) => a.question === q.question); return {

...q,

id: idx + 1,

userAns: attempt?.userAns || null, feedback: attempt?.feedback || null, rating: attempt?.rating || null,

correctAns: q.answer, // Corrected: Using "answer" from the JSON response instead of a separate column

isAttempted: !!attempt,

};

});

setMergedList(merged);

setLoading(false); // Set loading to false once data is fetched

};

const attemptPerc =

totalQuestions > 0 ? Math.round((attemptedQuestions / totalQuestions) \* 100) : 0;

const downloadReport = async () => { setIsGenerating(true); setOpenAll(true); // Expand all sections

await new Promise(resolve => setTimeout(resolve, 500)); // Wait for React rerender const element = feedbackRef.current;

// Hide buttons before generating

const buttonsToHide = element.querySelectorAll('.hide-in-pdf'); buttonsToHide.forEach(button => {

button.style.display = 'none'; button.style.visibility = 'hidden';

});

const opt = {

margin: 0.2, // inches

filename: 'Feedback\_Report.pdf', image: { type: 'jpeg', quality: 0.98 }, html2canvas: { scale: 2, useCORS: true },

jsPDF: { unit: 'in', format: 'a4', orientation: 'landscape' },

pagebreak: { mode: ['avoid-all', 'css', 'legacy'] }, // handle page breaks properly

};

html2pdf().from(element).set(opt).save().then(() => {

toast.success("PDF Downloaded Successfully!", { position: "bottom-right" }); setIsGenerating(false);

setOpenAll(false);

}).catch(err => {

console.error('PDF generation error:', err);

toast.error("Failed to generate PDF", { position: "bottom-right" }); setIsGenerating(false);

setOpenAll(false);

}).finally(() => {

// Restore buttons after PDF generation buttonsToHide.forEach(button => { button.style.display = ''; button.style.visibility = '';

});

});

};

{isGenerating && (

<div style={{

position: 'fixed', top: 0, left: 0, width: '100%', height: '100%',

backgroundColor: 'rgba(0, 0, 0, 0.5)', display: 'flex', alignItems: 'center', justifyContent: 'center', zIndex: 9999

}}>

<div style={{ textAlign: 'center', color: 'white' }}>

<div className="loader"></div>

<p style={{ marginTop: '20px', fontSize: '18px' }}>Generating PDF...</p>

</div>

</div>

)}

return (

<div className="max-w-screen-xl mx-auto px-6 sm:px-12 py-10">

<ToastContainer />

{loading ? (

<div className="space-y-4">

<Skeleton height={30} />

<Skeleton height={20} count={4} />

<div className="flex space-x-2">

<Skeleton circle width={40} height={40} />

<Skeleton circle width={40} height={40} />

</div>

</div>

) : mergedList.length === 0 ? (

<div className="text-center text-gray-500 text-xl font-medium"> No Interview Feedback Record Found

</div>

) : (

<>

{/\* Page Header \*/}

<div ref={feedbackRef} className="space-y-8 p-6 rounded-lg ">

<div className="report-header mb-10 p-6 border rounded-xl shadow-md bg-gray-50">

<h1 className="text-3xl font-bold text-center text-primary mb-6 tracking-wide"> Interview Feedback Report

</h1>

<div className="grid grid-cols-2 gap-4 text-base text-gray-800">

<div className="flex flex-col">

<span className="font-semibold">Candidate ID:</span>

<span>{userDetails.createdBy}</span>

</div>

<div className="flex flex-col">

<span className="font-semibold">Mock ID:</span>

<span>{userDetails.mockId}</span>

</div>

<div className="flex flex-col">

<span className="font-semibold">Position:</span>

<span>{userDetails.jobTitle}</span>

</div>

<div className="flex flex-col">

<span className="font-semibold">Years of Experience:</span>

<span>{userDetails.experience}</span>

</div>

</div>

</div>

<div className="report-header mb-10 p-6 border rounded-xl shadow-md bg-gray-50">

<header className="text-center mb-8 space-y-2">

<h1 className="text-4xl font-bold text-green-600"> Congratulations!</h1>

<h2 className="text-2xl font-semibold text-gray-800"> Here is your Full Interview Analysis

</h2>

<p className="text-gray-500 text-sm">

Below are all questions in sequence, with your performance metrics and correct answers.

</p>

</header>

{/\* Buttons Row at the top \*/}

<div className="flex justify-center gap-6 mb-12">

<Button

onClick={() => router.replace("/dashboard")}

className="px-6 py-3 text-base font-medium rounded-xl hide-in-pdf"

>

⬅ Go to Dashboard

</Button>

<Button onClick={downloadReport}

className="px-6 py-3 text-base font-medium rounded-xl bg-blue-600 text-white hide-in-pdf"

>

Download Report

</Button>

</div>

{/\* Summary Cards \*/}

<div className="flex flex-col md:flex-row justify-center gap-8 mb-12">

{/\* Attempt % \*/}

<div className="w-40 p-4 bg-white rounded-2xl shadow flex flex-col items-center">

<div className="w-20 h-20 mb-2">

<CircularProgressbar value={attemptPerc} text={`${attemptPerc}%`} styles={buildStyles({ textSize: "16px",

pathColor: getColorForPercentage(attemptPerc), textColor: getColorForPercentage(attemptPerc), trailColor: "#e5e7eb",

})}

/>

</div>

<span className="mt-2 text-sm font-semibold">Attempt %</span>

</div>

{/\* Avg Rating \*/}

<div className="w-40 p-4 bg-white rounded-2xl shadow flex flex-col items-center">

<div className="w-20 h-20 mb-2">

<CircularProgressbar value={averageRating \* 10} text={`${averageRating}`} styles={buildStyles({

textSize: "16px",

pathColor: getColorForPercentage(averageRating \* 10), textColor: getColorForPercentage(averageRating \* 10),

trailColor: "#fef3c7",

})}

/>

</div>

<span className="mt-2 text-sm font-semibold">Overall Rating</span>

</div>

</div>

{/\* Question-by-Question Feedback \*/}

<section className="space-y-6">

{mergedList.map((item, index) => (

<Collapsible key={index}

open={openAll ? true : undefined} // force open when needed

className={` border rounded-2xl shadow-md transition hover:shadow-lg p-4 my-4

${item.isAttempted ? 'bg-white border-l-4 border-green-400' : 'bg-white border-l-4 border-red-400'}

`}

>

<CollapsibleTrigger className="w-full text-left font-semibold text-lg flex justify-between items-

center">

{`Q${index + 1}: ${item.question}`}

<ChevronsUpDown className="ml-2 h-5 w-5" />

</CollapsibleTrigger>

<CollapsibleContent className="mt-4 space-y-4">

{item.isAttempted ? (

<>

{/\* Rating Section \*/}

<div className="flex flex-col items-center space-y-2">

<span className="font-bold text-gray-700">Rating:</span>

<div className="w-24 h-24">

<CircularProgressbar

value={item.rating ? parseFloat(item.rating) \* 10 : 0} text={`${item.rating || 0}`}

styles={buildStyles({ textSize: "16px",

pathColor: getColorForPercentage(item.rating \* 10), textColor: getColorForPercentage(item.rating \* 10), trailColor: "#e5e7eb",

})}

/>

</div>

</div>

{/\* Your Answer \*/}

<div className="p-4 rounded-xl border-2 border-red-500 bg-red-100 text-red-700">

<strong>Your Answer:</strong> {item.userA

ns || "No Answer Given"}

</div>

{/\* Correct Answer \*/}

<div className="p-4 rounded-xl border-2 border-green-500 bg-green-100 text-green-700">

<strong>Correct Answer:</strong> {item.correctAns}

</div>

{/\* Feedback \*/}

<div className="p-4 rounded-xl border-2 border-blue-500 bg-blue-100 text-blue-700">

<strong>Feedback:</strong> {item.feedback || "No feedback available"}

</div>

</>

) : (

<>

{/\* Unattempted \*/}

<div className="text-yellow-500 font-semibold"> Unattempted

</div>

{/\* Correct Answer \*/}

<div className="p-4 rounded-xl border-2 border-green-500 bg-green-100 text-green-700">

<strong>Correct Answer:</strong> {item.correctAns}

</div>

</>

)}

</CollapsibleContent>

</Collapsible>

))}

</section>

</div>

</div>

</>

)}

</div>

);

}

export default Feedback;

# Chapter 5: Testing

## Chapter5:Testing

### Test Strategy

The testing phase is crucial in ensuring the stability, accuracy, and performance of the JARVIS platform. It focuses on validating both functional and non-functional aspects of the system.

#### Objectives

* + - * Ensure all modules perform as expected under different scenarios.
      * Identify and fixbugs in individual components before integration.
      * Validatedata integrity and AI response accuracy.

#### Testing Levels

1. **Unit Testing**
   1. Conducted on individual modules like InterviewForm, RecordAnswerSection, and FeedbackList.
   2. Ensuresisolated logic (e.g., API calls, form validations, speech recognition) works correctly.

#### Integration Testing

* 1. Checksflowacrosscomponents: Sign In → Create Interview→ Answer → Feedback.
  2. Ensures smooth interaction between Clerk authentication, Gemini API, Drizzle DB, and Frontend.

#### System Testing

* 1. End-to-end tests simulate a complete interview flow.
  2. Includes checking audio/video inputs, dynamic question generation, and feedback accuracy.

#### User Acceptance Testing (UAT)

* 1. Performed with sample users (students, mentors).
  2. Ensuresthat UI is intuitive and output is meaningful and actionable
  3. **Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test CaseID** | **Test Scenario** | **Input** | **Expected Output** | **Status** |
| TC\_001 | Userloginvia Clerk | Valid email/password | Redirect to Dashboard | Pass |
| TC\_002 | Add new interview session | Jobrole: Full Stack,2+ years exp. | AIquestionsgenerated, interview created | Pass |
| TC\_003 | Startinterview with speech- to-text | Voice input | Answersaved, feedback generated | Pass |
| TC\_004 | AIfeedback generation | Answerto technical question | Feedback JSON with ratingand suggestions | Pass |
| TC\_005 | Display feedback UI | Collapse/expand interaction | Questionsshownwith correct answer and rating fields | Pass |
| TC\_006 | Reviewpast sessions | Go to Interview List | Previousinterviews displayed with feedback/start option | Pass |
| TC\_007 | Use text-to- Speech for question | Question click on | System speaks question | Pass |
| TC\_008 | Sessiondata persistence | Refreshinterview screen | Questionstate and responses retained | Pass |
| TC\_009 | Invalidinput handling | Submitblankjob role | Validation error shown | Pass |
| TC\_010 | Gemini API fail scenario | API key missing | Errorlogged, useralerted | Pass |

# Chapter 6: Limitations of Proposed System

## Chapter 6: Limitations of Proposed System

Despite offering arobust AI-driven mockinterview experience, thecurrentversion of JARVIS has certain limitations that can be addressed in future iterations:

### AI Feedback Dependence on Gemini API

* + - Thesystem heavily depends on Gemini 2.0 Flash API for generatingquestionsand evaluating user responses.
    - If the API service is interrupted or deprecated, the application loses its core NLP functionality.
    - Ratelimitingor APIkey misuse can cause failures in real-time feedback generation.

### LimitedReal-Time Biometric Monitoring

* + - While webcamsupport is available, the system **does not yet perform live facial expression analysis or emotion detection** using tools like OpenCV.
    - Non-verbal cue analysis (eye contact, facial stress) is not fully implemented, limiting behavioral insight.

### Limited Question Personalization

* + - Although questionsare generated based on role, experience, and job description,

#### they do notyetaccount for user historyor skillgaps.

* + - No adaptive difficulty scaling or personalized learning pathsare available at this stage.

# Chapter 7: Proposed Enhancements

## Chapter 7: Proposed Enhancements

To evolve JARVIS into a more powerful, realistic, and insightful mock-interview platform, the following enhancements are proposed. Each enhancement addresses a current limitation (see Chapter 6) and adds significant value for end users.

### Advanced Biometric and Behavioral Analytics

**Objective:** Capturenon-verbal cues to mirrorin-person interviews.

#### Facial Expression Analysis

* + - * Integrate **OpenCV** or MediaPipeto track eye contact, smilefrequency, and micro- expressions.
      * Provide metrics on “engagement score” and “confidence level.”

#### Posture s Gesture Monitoring

* + - * Usepose-estimation libraries to detect slouching, fidgeting, or confident stance.
      * Offer actionable tips (e.g. “sit upright,” “maintain steady gaze”).

#### Voice Modulation Analysis

* + - * Analyze tone, pitch variation, and speech ratevia Web Audio API.
      * Generate “vocal impact” score and suggestions (e.g. “slow down,” “add emphasishere”)..

### Adaptive Question Difficulty

**Objective:** Personalize difficulty based on user performance.

#### Skill Profiling

* + - * Maintain ahistory of user scores by topic (e.g. Data Structures, System Design).
      * Identify weak areasviaanalytics.

#### Dynamic Prompt Adjustment

* + - * Adjust AI prompt to generateeasier or harder follow-up questions.
      * Implement “level-up” mechanism: when user consistently scores ≥8/10, increase complexity.

### Mobile-Friendly s Offline Support

**Objective:** Extend accessibility to mobile devices and low-connectivity scenarios.

#### Responsive PWA

* + - * Convert JARVIS into a Progressive Web App foroffline cachingof question sets.
      * Enable users to practice without continuous Internet access.

#### Local Storage Sync

* + - * Storeintermediateanswersin IndexedDB when offline; sync to server when reconnected.

## Multi-Role and Panel Interview Simulation

**Objective:** Emulate real panel interviews with multiple AI “interviewers.”

#### Multi-Actor Questioning

* + - * Createmultiple AI agents (e.g. HR, Technical Lead) withdistinct prompt personas.
      * Rotate questioning styles: behavioral, technical, situational.

#### Panel Feedback Aggregation

* + - * Collect and display feedback from each AI persona separately.
      * Provide consolidated “panel summary” and individual “panelist scores.”

### 7.4 Integration withReal-World Platforms

**Objective:** Bridge JARVISwith industrytools andlearningresources.

#### LMS Integration

* + Connectwith Learning Management Systems (Moodle, Canvas) to track student progress.

#### Job Portal APIs

* + Fetch real job descriptions from LinkedIn or Indeed to generate up-to-date, role-specific questions.

# Chapter 8: Conclusion

## Chapter8:Conclusion

The JARVISprojecthas provento be an innovativeanduseful AI-powered mockinterview platform that assists users in overcoming interview anxiety and improving their performance. By simulating real-world interview scenarios, JARVIS empowers students, job seekers, and professionals to practice their interview skills in a controlled, intelligent environment.

### Achievements of the JARVISProject

* + - **AI-Powered Question Generation:** Theintegration ofthe Gemini APIenables dynamic, role-specific questions, providing users with tailored interview preparation.
    - **Speech Recognition s Feedback:** The inclusion of speech-to-text technology allows for a realistic interaction with the system, and the AI provides detailed feedback based on fluency, relevance, and accuracy.
    - **Real-Time Feedback Mechanism:** Users receive immediate feedback on their performance, making it easier to track their improvement and identify areas for development.
    - **Personalized Learning Experience:** The system's ability to generate questions based on job roles, job descriptions, and years of experience ensures that each user receives a relevant and unique experience.

### Future Scope of JARVIS

While the current version of JARVIS providessignificant value, thereis always room for improvement and future expansion:

* + - **Behavioral and Non-Verbal Cues Analysis:** Integrating facial expression and gesture analysis will provide more comprehensive feedback, simulating a real interview environment.
    - **Mobile Support s Offline Access:** Expanding the platform to mobile devices and supporting offline modes will increase accessibility for users across diverse locations and situations.

### Conclusion Summary

JARVIS represents a significant step forward in transforming the traditional interview preparation process. By leveraging AI, speech recognition, and dynamic question generation, the system offers a personalized, intelligent, and realistic mock interview experience. As the project evolves, incorporating additional features such as biometric analysis and advanced data visualizations will enhance the platform’s ability to meet the needs of its users. The future enhancements outlined in this report aimto make JARVIS not just a mock interview tool, but a comprehensive interview coaching platform that can guide users through every stage of their preparation, offering tailored feedback and performance trackin

# Chapter 9: Bibliography

## Chapter 9: Bibliography

### Documentation and Online Resources

1. **Gemini API Documentation**, Google Cloud
   1. Guided the integration of Gemini AIfor question generationandfeedback.

#### React Documentation

* 1. Essential for component-based development and state management.

#### Drizzle ORM Documentation

* 1. Assisted in managing the database schema and interactions with Neon DB.

#### Clerk Authentication Documentation

* 1. Implemented secure authentication for users in JARVIS.

#### react-hook-speech-to-text

* 1. Enabled speech recognition for answeringinterview questions.

#### Tailwind CSS Documentation

* 1. Used for responsive UI design in thefrontend.

#### OpenCV Documentation

* 1. Helped integrate facial recognition andimage processing (future enhancement).

#### GitHub Repositories for AI Models and Tools

* 1. Referenced for model integration and API setup for AI functionalities.

### Online Articlesand Blogs

#### "How to Builda Machine Learning Model with OpenCVand Python"

* 1. Practical approach to OpenCV integration for future facial expression analysis.

#### "Getting Started with Serverless Databases: Neon"

* 1. Guide to Neon serverless database setup, used in JARVIS backend.

#### "Buildinga Scalable Web Application with React and Node.js"

* 1. Insights on integrating React frontendwith Node.jsbackend.

### Miscellaneous Sources

#### YouTube Tutorialson Speech Recognition

* 1. Provided practical examples for integrating speech-to-text functionality.

#### Stack Overflow Discussions

* 1. Community-driven solutionsfor React, Node.js, and AI model integration issues.

Theseresources were crucial forthe development of JARVIS, providingfoundational knowledge and practical implementation guidance throughout the project.

# Chapter 10: User Manual

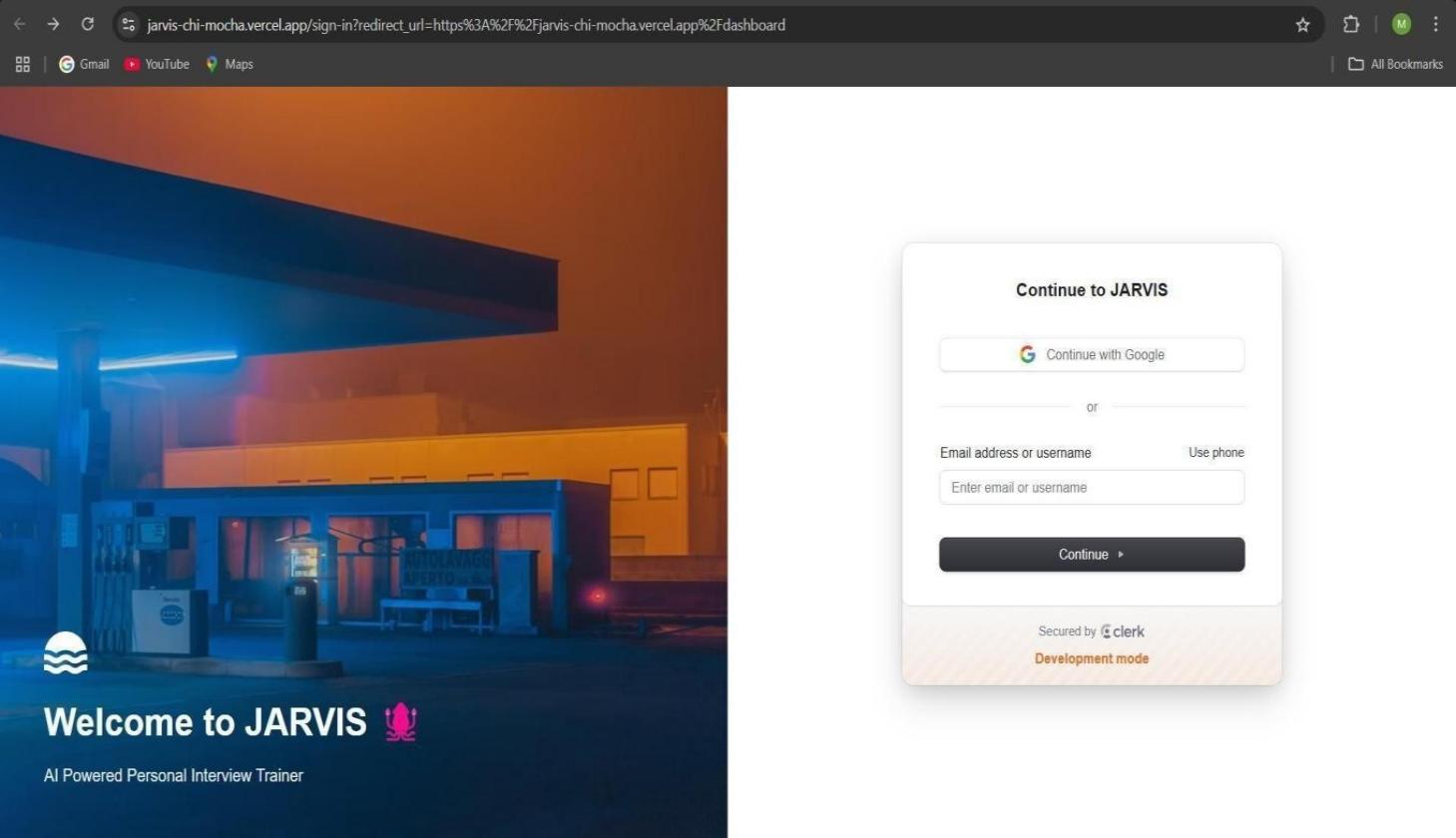
### Login / Sign Up Screen

#### Purpose:

Allows users to securely authenticate into the system using Clerk authentication (Email or Social Login).

#### Validations:

* + - 1. Email format is validated (must be a valid email address).
      2. Password policies (if used) enforced by Clerk.
      3. Mandatory to complete sign-in to access any feature.



### Dashboard

#### Purpose:

Displays user-specific actions like creating a new interview or viewing past interviews.

#### Description: Contains two cards:

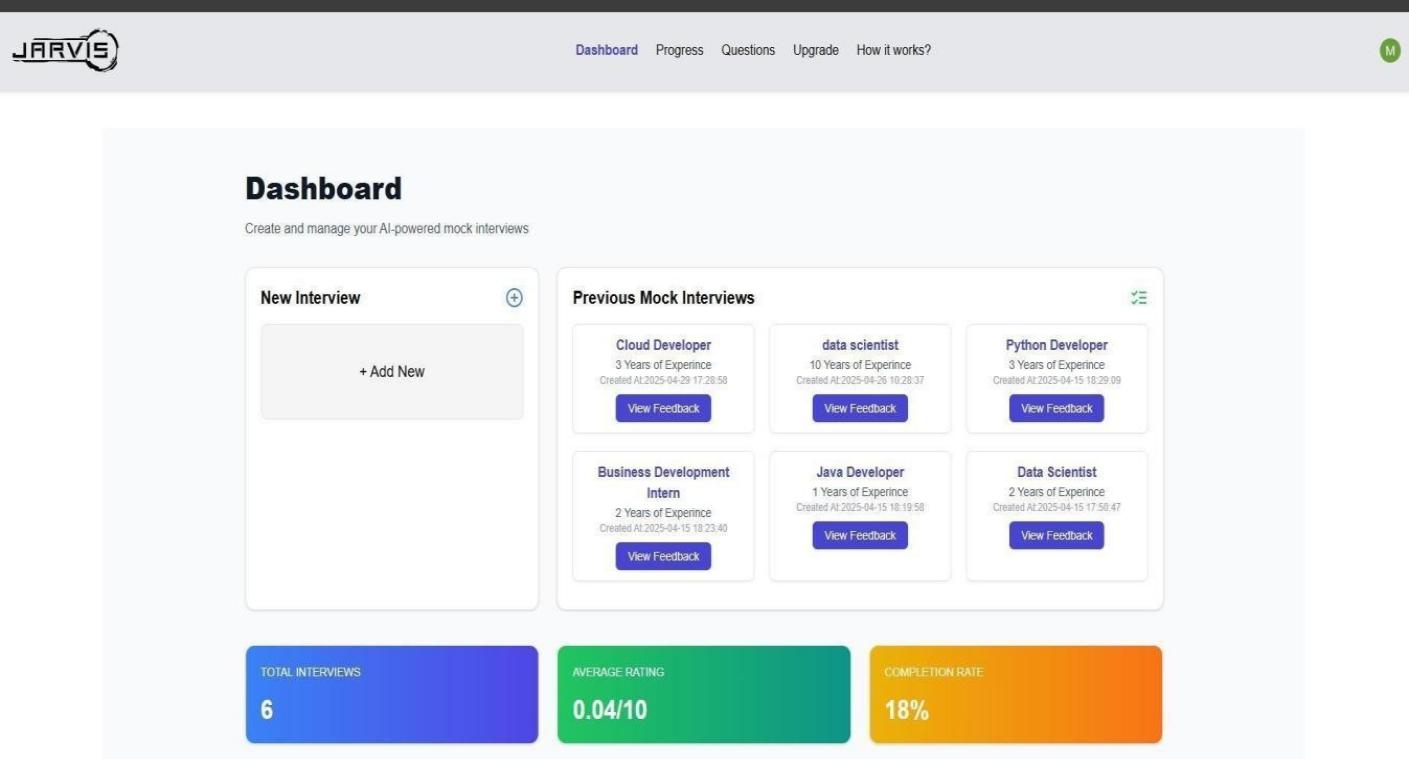
* + - 1. New Interview → Launches form to create a new mock session.
      2. Past Interviews → Displays previously taken interviews.

#### Also displays:

1. Total Interviews
2. Average Rating
3. Completion Percentage

#### Validations:

No manual data input; cards are auto-populated dynamically based on the database.



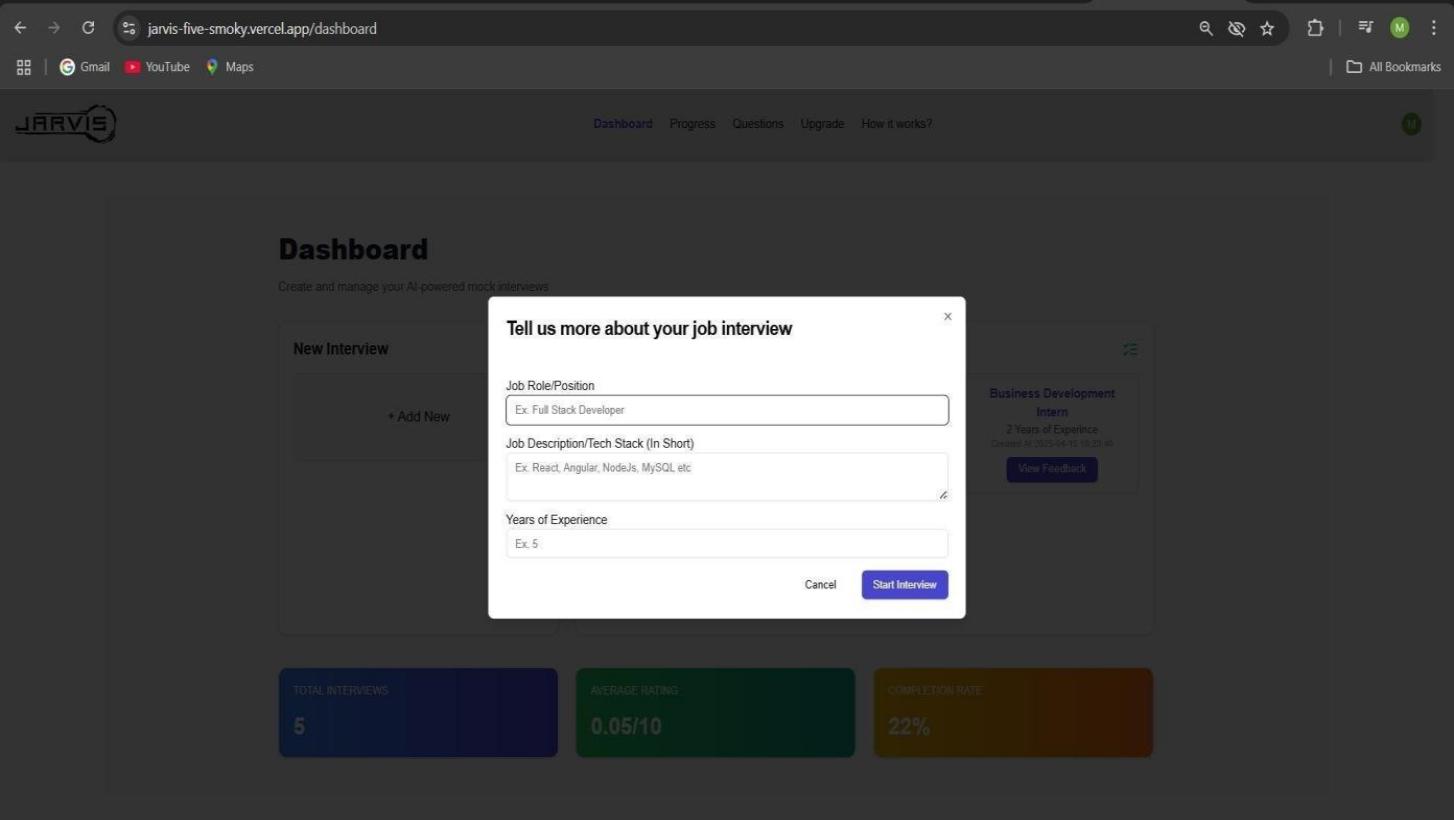
### Add New Interview Screen

#### Purpose:

Allows user to input interview context to generate personalized AI interview question

#### Fields & Validations:

* + - 1. **Job Title**: Required, min 3 characters.
      2. **Job Description:** Required, min 15 characters.
      3. **Years of Experience**: Required, numeric, between 0–30.



### Questions Page

#### Purpose:

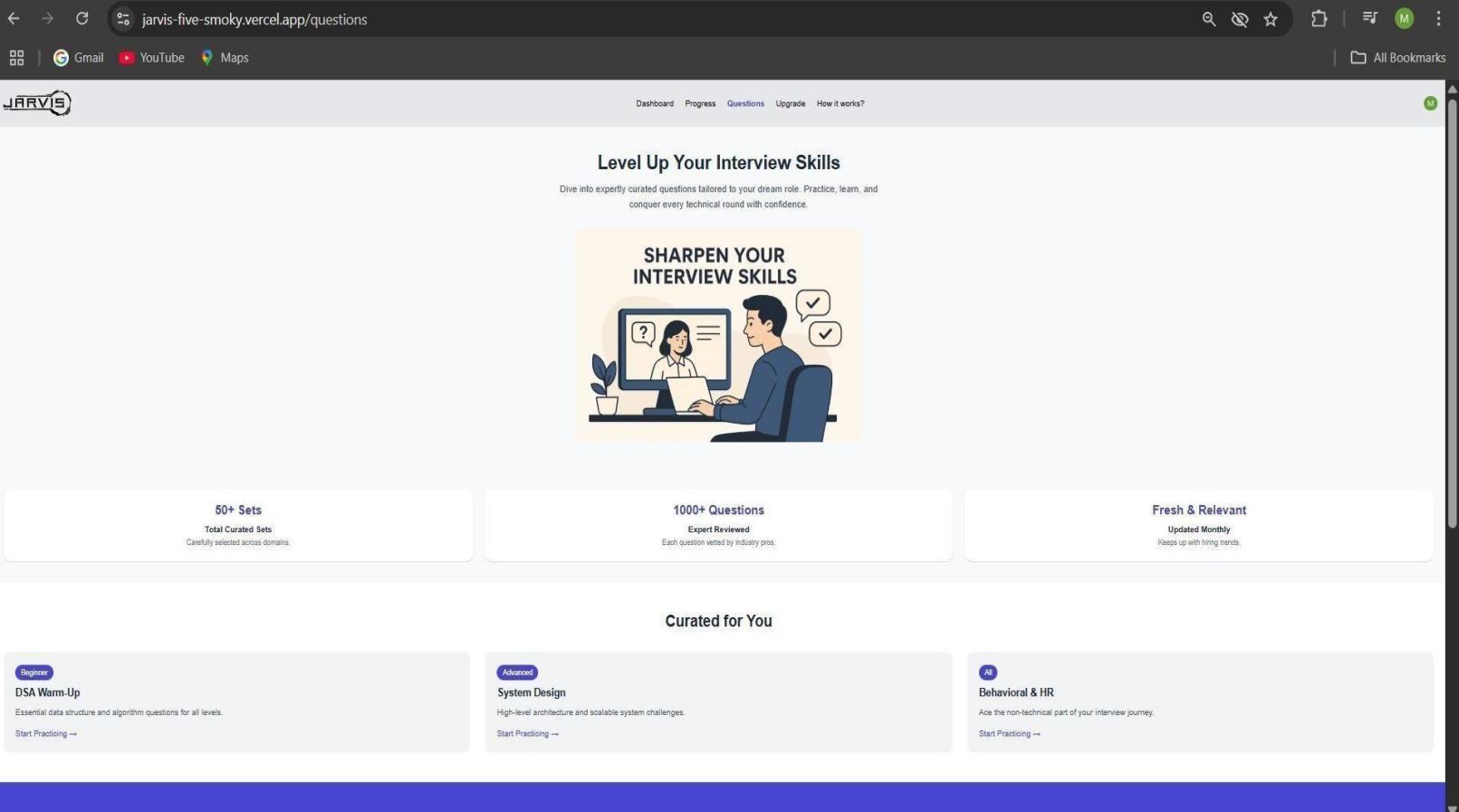
Shows curated question sets based on topic: HR, DSA, System Design, etc.

#### Description:

* + - 1. Clicking a set navigates to the AI mock session.
      2. Provides resource links for preparation.

#### Validations:

* + - 1. No user input, only click actions.
      2. Role-based access is applied to premium content.



;

### Interview Screen (Mock Session)

#### Purpose:

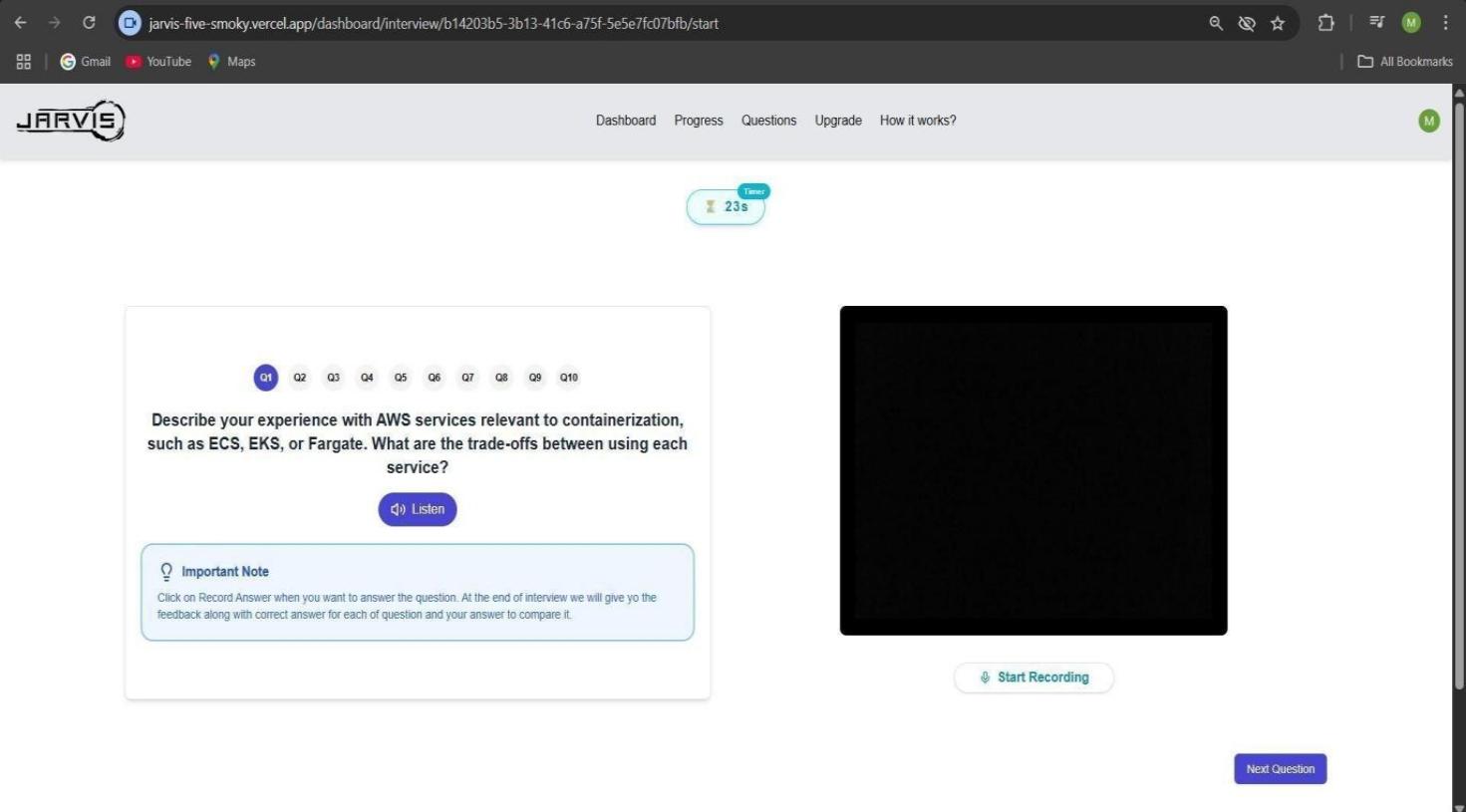
Main interface for answering interview questions with recording & timer.

#### Description:

* + - 1. Questions appear one by one.
      2. Timer (e.g., 60 seconds) for each question.
      3. Record Answer button records audio/video.
      4. Next and Previous buttons to navigate.
      5. End Interview button only enabled after min 3 questions attempted.

#### Validations:

* + - 1. Timer enforces answering limit.
      2. User cannot return to previous questions after timeout.
      3. Cannot end interview unless minimum attempt threshold met.
      4. Audio must be captured before proceeding (optionally enforced).



### Feedback Page

#### Purpose:

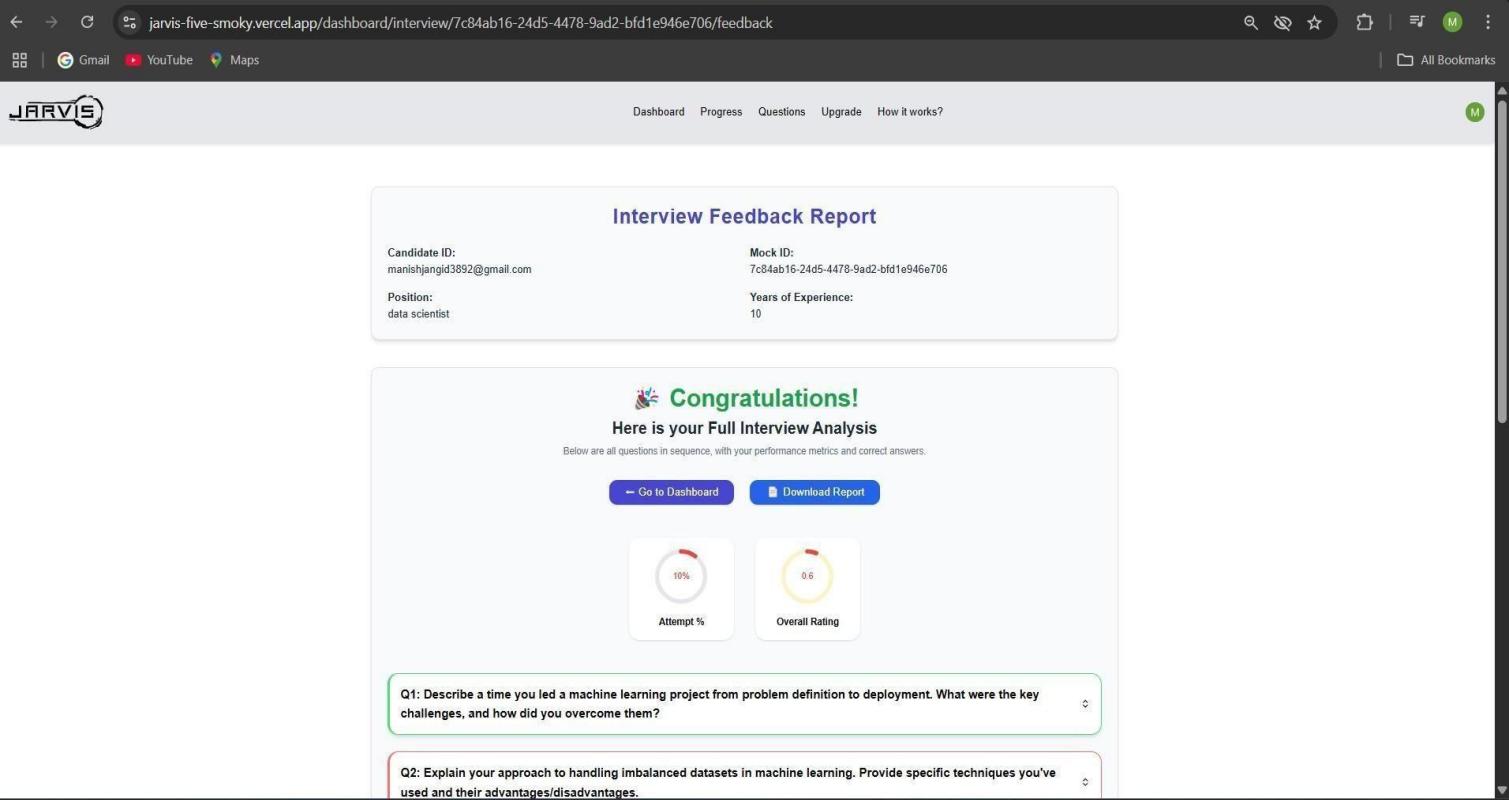
Shows AI-generated analysis post-interview.

#### Description:

* + - 1. Lists all questions in collapsible format.
      2. For each:
         1. User Answer
         2. Correct Answer
         3. AI Feedback
         4. Rating
      3. Also shows:
         1. Overall Rating (Graph)
         2. Attempt Percentage (Graph)

#### Validations:

* + - 1. If a question is unattempted, user answer and feedback are shown as "Not attempted".
      2. Rating bars change color based on performance.



### Progress Page

#### Purpose:

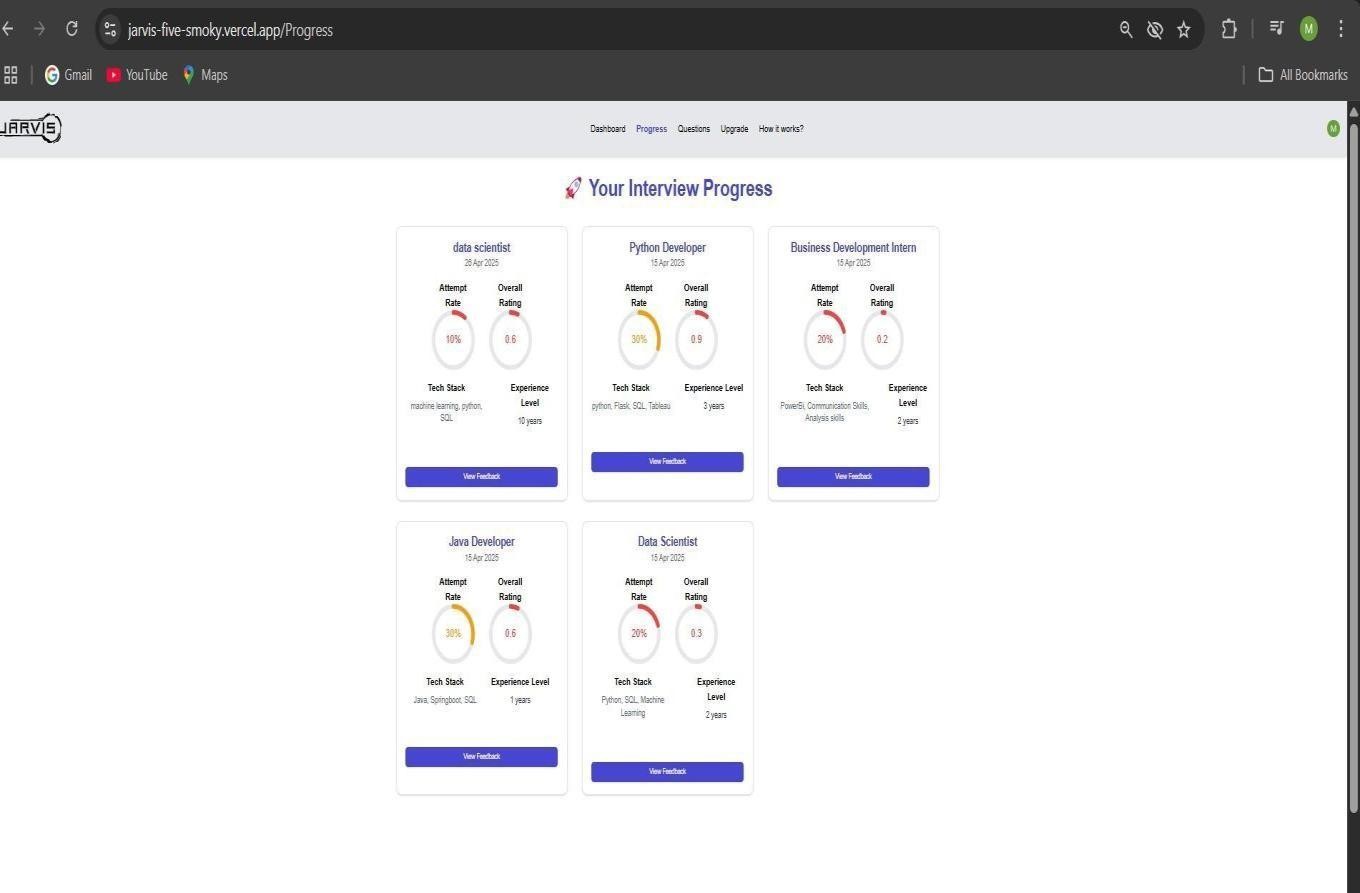
Shows analytics for all interviews.

#### Description:

* Card for each interview with:
  1. Job Position, Date
  2. Completion %
  3. Average Rating
  4. Tech Stack
  5. Experience Level
  6. View Feedback Button

#### Validations:

If no interviews found, a loading animation followed by "No progress found" message is shown.



### Upgrade Page (Optional)

#### Purpose:

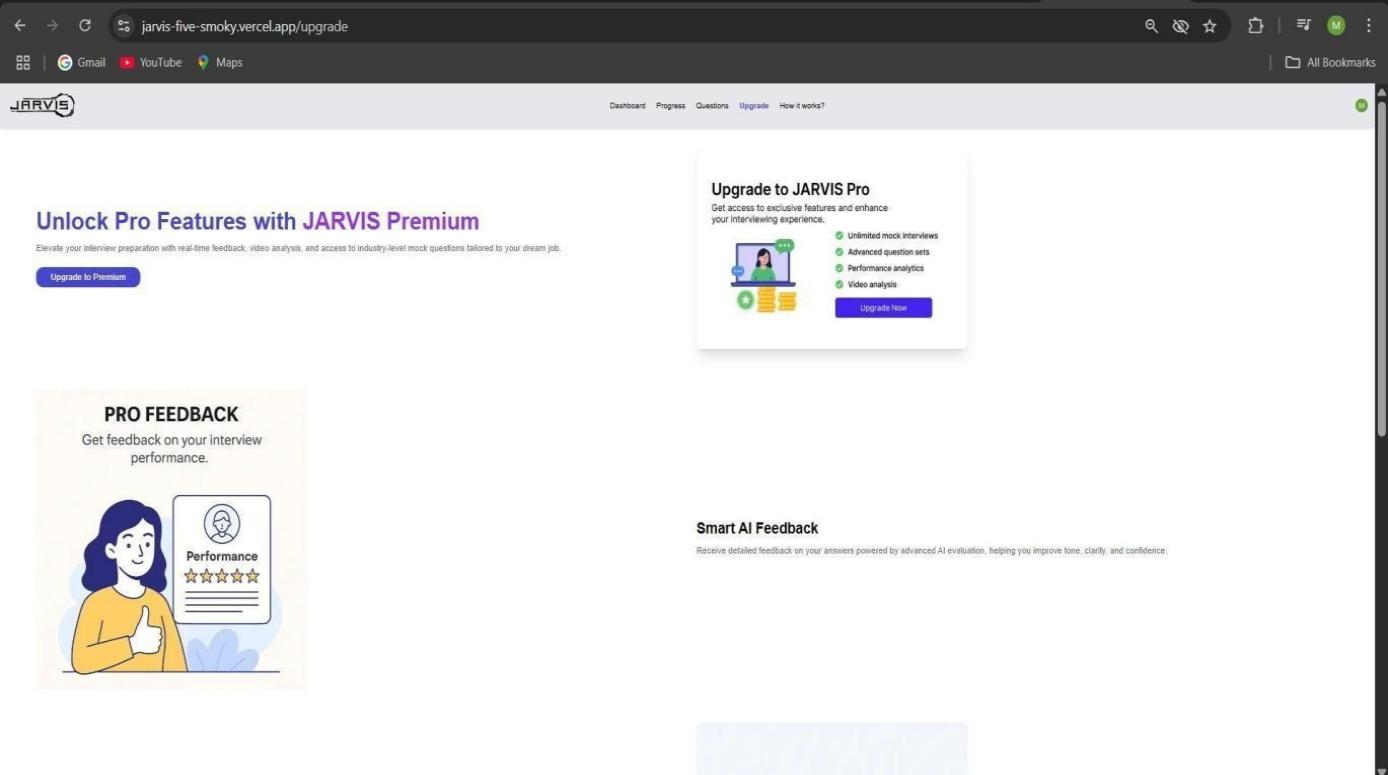
Promotes premium features.

#### Description:

Lists benefits: AI Feedback, Video Analysis, Premium Question Sets, etc.

#### Validations:

Button to Upgrade (Currently non-functional or linked to Stripe/Upgrade plan).



### How It Works Page

#### Purpose:

Educates users on the JARVIS workflow.

#### Description:

* + - 1. Step-by-step explanation:
      2. Speak Answer
      3. AI Feedback
      4. Review & Improve

#### Validations:

Static content; no validation needed.

