Chapter 1
Introduction

Chapter 2
Software &Hardware Requirements

Chapter 3
Problem Description

Chapter 4
Literature Survey

Chapter 5
Software Requirements Specification

Chapter 6
Software and Hardware Design

Chapter 7
Web Module

Chapter 8
Coding

Chapter 9
Result and Screen Output

Chapter 10
Conclusion and Future Work

AI-POWERED INTELLIGENT INTERVIEW AND EVALUATION PLATFORM

A

MAJOR PROJECT - I REPORT

Submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

GROUP NO. 49

Yash Yadav	**0187CS221219**
Raj Mahajan	**0187CS221159**

Under the guidance of

Dr. Amit Dubey

(Associate Professor)

Department of Computer Science & Engineering
Sagar Institute of Science & Technology (SISTec), Bhopal (M.P)

Approved by AICTE, New Delhi & Govt. of M.P.
Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

December-2025

Sagar Institute of Science & Technology (SISTec), Bhopal (M.P)

Department of Computer Science & Engineering

CERTIFICATE

We hereby certify that the work which is being presented in the B.Tech. Major Project-I Report entitled **AI-POWERED INTELLIGENT INTERVIEW AND EVALUATION PLATFORM** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology***, submitted to the Department of **Computer Science & Engineering**, Sagar Institute of Science & Technology (SISTec), Bhopal (M.P.) is an authentic record of our own work carried out during the period from Jul-2025 to Dec-2025 under the supervision of Prof. Amit Dubey.

The content presented in this project has not been submitted by me for the award of any other degree elsewhere.

Yash Yadav	**Raj Vijay Mahajan**
0187CS221218	**0187CS221224**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Amit Dubey	**Prof. Nargish Gupta**	**Dr. Manish Billore**
Project Guide	**HOD, CSE**	**Principal**

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. Manish Billore**, **Principal, SISTec and Dr. Swati Saxena, Vice Principal SISTec** Gandhi Nagar, Bhopal for giving us an opportunity to undertake this project.

We also take this opportunity to express a deep sense of gratitude to **Prof. Nargish Gupta, HOD, Department of Computer Science & Engineering** for his kindhearted support.

We extend our sincere and heartfelt thanks to our guide, **Dr. Amit Dubey,** for providing us with the right guidance and advice at crucial junctures and for showing us the right way.

I am thankful to the **Project Coordinator, Prof. Deepti Jain,** who devoted her precious time in giving us the information about various aspects and gave support and guidance at every point of time.

I would like to thank all those people who helped me directly or indirectly to complete my project whenever I found myself in any issue.

TABLE OF CONTENT

ABSTRACT

“The AI-powered Intelligent Interview and Evaluation Platform Technical Interview Platform” focuses on improving the efficiency, structure, and accessibility of technical hiring through a unified digital system. The platform integrates real-time communication tools such as secure video calls, screen sharing, and session recording with an interactive coding environment, enabling seamless collaboration between interviewers and candidates. By combining live coding, synchronized interactions, and an organized interview workflow, the system minimizes technical disruptions and enhances the fairness and accuracy of candidate evaluations.

Aligned with **SDG 9 – Industry, Innovation, and Infrastructure**, this solution promotes innovation and supports modern digital recruitment practices by offering a reliable, scalable, and user-friendly platform for remote technical assessments.

LIST OF ABBREVIATIONS

Abbreviation	**Description**
SDLC	Software Development Life Cycle
SRS	Software Requirements
HTML	Hype Text Markup Language
UM	Unified Modelling Language
DB	Database
UI	User Interface
API	Application Programming Interface
UX	User Experience
MERN	Mongodb, Express.js, React.js, Node.js
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
JS	JavaScript
JSON	Java Script Object Notation

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 ABOUT PROJECT

AI-powered Intelligent Interview and Evaluation Platform is a real-time technical interview collaboration platform that integrates video calls, screen sharing, and recording with an interactive coding environment. It allows interviewers and candidates to collaborate efficiently during live sessions while solving coding and problem-solving tasks. The system supports Data Structures and Algorithm (DSA) problems and provides a secure, efficient, and scalable solution suitable for remote interviews.

By combining modern technologies such as Next.js, Convex Database, Clerk Authentication, and Stream Video SDK, AI-powered Intelligent Interview and Evaluation Platform enables smooth real-time communication, secure access, and data handling. This innovative platform aims to redefine the online interview process by offering a single, feature-rich, and user-friendly interface that enhances productivity, fairness, and collaboration.

1.2 PURPOSE

The primary purpose of the AI-powered Intelligent Interview and Evaluation Platform Platform is to create an efficient, interactive, and unified digital environment that enables interviewers and candidates to conduct technical interviews.The system aims to bridge the gap between traditional interview methods and modern digital hiring needs by offering a complete real-time collaboration platform for technical assessments.

It provides a structured space where interviewers can evaluate a candidate’s coding skills, problem-solving abilities, and communication in one integrated environment. The platform facilitates direct interviewer–candidate interaction through a 1-to-1 Video Call and Screen Sharing feature.

1.3 PROJECT OBJECTIVE

Build a unified web-based platform that enables interviewers and candidates to conduct technical interviews efficiently within a single integrated environment. Make remote interviews easily accessible to organizations and applicants through a user-friendly and interactive digital interface. Implement a 1-to-1 video call and screen-sharing feature that supports direct communication and enables clear, real-time assessment during technical rounds.

Develop an embedded Monaco-based coding editor that allows candidates to solve DSA problems while interviewers evaluate their approach and problem-solving skills. Provide a structured workspace that promotes active participation, smooth collaboration, and accurate technical evaluation.
The objective is to simplify the interview process, reduce dependency on multiple tools, and deliver a more organized, professional, and seamless technical interview experience through a unified digital system.

1.4 SIGNIFICANCES OF PROJECT

Helps connect aspiring professionals and students with experienced mentors, overcoming the limitations of traditional mentorship. Provides mentees with structured guidance and expert insights to support career growth and skill improvement.
Offers a digital platform that makes mentorship available anytime and anywhere, increasing convenience for both mentors and mentees. Enables effective communication through a 1-to-1 Video call feature, fostering better understanding and engagement.
Provides instant support and guidance through an integrated chatbot. Simplifies the overall mentorship process by combining communication, scheduling, and guidance within a single system. To support AI-driven mentor–mentee matching, automated scheduling, and a freemium model for improved efficiency, personalization, and sustainability.

CHAPTER 2
SOFTWARE AND HARDWARE REQUIREMENT

The successful development and implementation of the AI-POWERED INTELLIGENT INTERVIEW AND EVALUATION PLATFORM require specific software and hardware components. These components ensure smooth operation, scalability, and ease of use for all stakeholders.

2.1 SOFTWARE REQUIREMENTS

The AI-powered Intelligent Interview and Evaluation Platform Platform relies on a well-defined set of software requirements to ensure smooth development, deployment, and operation of the system. The platform's frontend is built using modern web technologies such as React.js and Next.js, providing a responsive, fast, and user-friendly interface for both interviewers and candidates. Styling is handled through Tailwind CSS and Radix UI, ensuring a clean and consistent user experience across devices.

The backend is powered by Convex, which provides real-time server logic, data synchronization, and API handling, allowing seamless interaction between users during live interviews. For secure authentication and user management, the system uses Clerk, offering a reliable and scalable identity layer for login and access control. Integrated communication features such as video calling, screen sharing, and session recording are enabled through the Stream Video SDK, ensuring low-latency and high-quality interaction. These software components work cohesively to deliver a unified, efficient, and scalable platform for conducting modern technical interviews.

2.2 HARDWARE REQUIREMENTS

The AI-powered Intelligent Interview and Evaluation Platform Platform requires reliable hardware infrastructure to support smooth development, deployment, and user accessibility. The development environment operates efficiently on systems equipped with at least an Intel Core i3 or equivalent processor, 4 GB RAM, and 256 GB SSD storage, ensuring responsive performance during coding, testing, and integration of real-time collaboration features.

For deployment, the platform benefits from server hardware that includes a quad-core processor, 8 GB RAM, and 512 GB storage to manage video call sessions, coding interactions, database operations, and multiple concurrent users. Stable, high-speed internet with a minimum bandwidth of 100 Mbps is essential to maintain uninterrupted video communication and low-latency data synchronization.

This hardware setup offers a scalable and dependable foundation for AI-powered Intelligent Interview and Evaluation Platform, meeting the performance needs of organizations while ensuring consistent accessibility and seamless operation across various platforms and devices.

2.3 HARDWARE REQUIREMENTS TABLE

Table 2.1: Hardware Requirements Table

Category	Hardware Components	Minimum Specifications	Purpose
Development System	Processor	Intel Core i3 or equivalent/AMD Ryzen 3	Efficient coding and development processes
	RAM	4GB or more	Basic multitasking during development
	Storage	128GBHDD or SSD	Store code, assets, and project files
	Operating System	Window10/ mac OS /Linux	Compatibility with development tools
Server Requirements	Processor	Dual-core CPU or higher	Basic server handling and user requests
	RAM	8GB or more	Manage backend processes and database
	Storage	256 GB HDD or SSD	Store platform data and user information
	Internet Connection	50 Mbps or higher	Stable server-client communication

CHAPTER 3
PROBLEM DESCRIPTION

In today's hiring environment, organizations conducting technical interviews often struggle due to the lack of a unified and efficient digital system. Interviewers typically rely on multiple disconnected platforms such as separate video conferencing tools, coding editors, and manual evaluation methods. This creates fragmented workflows, frequent interruptions, and difficulties in maintaining consistency during assessments. At the same time, candidates face challenges in switching between tools, dealing with technical issues, and presenting their skills effectively in an unstructured interview setup.

The AI-powered Intelligent Interview and Evaluation Platform Platform addresses this gap by providing a centralized digital solution that integrates live communication and coding assessments into a single interface. It streamlines the process of conducting technical interviews, enables smooth collaboration between interviewers and candidates, and ensures organized evaluation through real-time coding, screen sharing, and recording features. This approach helps organizations manage interviews efficiently while offering candidates a more structured and engaging experience during the hiring process.

3.1 CURRENT SCENARIO AND CHALLENGES

In the current scenario, technical interviews are conducted using separate and informal digital methods, such as external video meeting tools and independent coding platforms, leading to a fragmented and inefficient process. Interviewers struggle to maintain consistency and visibility while evaluating candidates because communication and coding take place on different applications.

Interviewers often struggle to observe the candidate's real-time problem-solving approach, and candidates find it challenging to present their skills effectively across disconnected tools. Managing communication, coding tasks, and session flow without a centralized system leads to confusion and delays. By addressing these issues, the platform aims to facilitate better communication, collaboration, and evaluation between interviewers and candidates, ensuring a more seamless and professional technical interview experience.

CHAPTER 4
LITERATURE SURVEY

Mentorship platforms combine social, educational, and technical elements to connect mentors and mentees for career and skill development. The literature spans theoretical foundations (control, algorithms, human–computer interaction), software engineering best practices, and practical web/mobile platform design. This survey synthesizes core findings relevant to building a robust, user-centric mentorship system [1][2].

Early mathematical and control-theory work (e.g., convolution operator applications and optimal control frameworks) show how rigorous formal methods can model and optimize system behavior in engineered systems [3]. While these papers are not about mentoring platforms directly, they illustrate useful methodologies for designing algorithms that must satisfy constraints (minimizing latency, optimizing scheduling or resource allocation) concepts applicable to mentor–mentee matching and scheduling subsystems where optimality and constraints matter [4].

Traditional Classic software engineering texts emphasise modularity, maintainability, testing and lifecycle practices. For distributed web platforms (storage, video, scale), references on storage networks and architecture underline the need for scalable, fault-tolerant storage and clear data-flow designs. That body of work supports choosing robust backend architectures and careful API/DB design [5].

Implication for Mentor Connect Adopt proven software engineering patterns: separation of concerns (auth, payment, matching, video), versioned APIs, automated testing, and clear data/schema design for mentors/mentees [6].

Human–computer interaction literature stresses user centered and participatory design to increase adoption and engagement. Usability research suggests that mentorship systems must make onboarding frictionless, display trust signals (profiles, endorsements), and provide clear workflows for booking and mentoring sessions. Interaction design resources further show that consistent, accessible UI/UX increases retention and reduces cognitive load for users.

Research on recommendation systems applied to career guidance suggests hybrid approaches (content-based + collaborative filtering + rule-based constraints) work best when data is sparse or there are cold-start users [7][10].

For mentorship, semantic matching on skills, goals, and experience level plus soft constraints (time zones, language) yields better mentor–mentee fit than purely popularity-based matching. Where specialized research is lacking in the references you shared, practical engineering docs and platform examples (course/mentorship sites) provide useful heuristics. Integrating video calls and chat requires attention to low-latency media transport, session security, and handling bandwidth variability. Practical platform docs (e.g., Firebase for auth and signaling; WebRTC/meeting standards for media) recommend using proven services or libraries for authentication, real-time signaling, and media handling rather than building low-level media stacks from scratch. Privacy and secure data handling (encryption in transit, access controls) are non-negotiable for user trust [8][9].

Contemporary web development best practices call for using modern, supported frameworks: React for frontend development, Node.js for backend microservices, and MongoDB/Mongoose for flexible schema modelling when user profiles have varied attributes. These tools speed development and integrate well with cloud services for scaling. Direct interaction is made possible through an embedded video conferencing system, enabling real-time, personalized communication between mentors and mentees. This feature enhances engagement, clarity, and trust while simulating an in-person mentoring experience. Future versions of the platform will leverage artificial intelligence to automate mentor–mentee matching based on interests, skills, goals, and previous interactions. Additionally, automated scheduling will optimize session planning, reducing manual effort and improving time management for both mentors and mentees [11].

The literature and practical engineering resources indicate that building a successful mentorship platform requires combining strong software engineering practices, careful UX design, principled algorithm choices, and rigorous privacy/security practices. Mentor Connect is well-positioned to apply these lessons: begin with a robust technical foundation, build transparent trust mechanisms, and collect outcome data to iteratively improve matching and user experience [12].

CHAPTER 5
SOFTWARE REQUIREMENT SPECIFICATION

5.1 INTRODUCTION

This chapter describes the Software Requirements Specification (SRS) for the **AI-powered Intelligent Interview and Evaluation Platform Real-Time Technical Interview Platform**. It defines the system's purpose, functional and non-functional requirements, performance expectations, and operational constraints.AI-powered Intelligent Interview and Evaluation Platform is designed to streamline the technical interview process by integrating features such as **live video calling, interactive coding environments, screen sharing, session recording**, and role-based access. The platform supports interviewers and candidates in conducting structured, smooth, and efficient remote interviews.

5.2 FUNCTIONAL REQUIREMENTS

Table 5.2: Functional Requirements

S. No.	Module	Functional Requirement Description
1	Authentication	Provides secure login and registration using Clerk authentication for Interviewer and Candidate roles.
2	Interview Session Management	Allows interviewers to create, schedule, and manage interview sessions.
3	Real-Time Communication	Supports video calls, screen sharing, and audio communication using Stream SDK.
4	Coding Environment	Provides a real-time collaborative Monaco Editor for coding interviews.
5	Recording Module	Records interview sessions (audio, video, and code activity) for later review.

5.3 NON-FUNCTIONAL REQUIREMENTS

Table 5.3: Non-Functional Requirements

Category	Requirement Description
Performance	The platform should load pages within 2–3 seconds and support real-time communication without delays.
Scalability	Must handle multiple concurrent interview sessions without affecting performance.
Security	Authentication, encrypted database entries, role-based access, and secure API communication are required.
Reliability	The system should maintain stable uptime with error handling and fallback mechanisms.
Usability	The interface must be simple, intuitive, and accessible for both technical and non-technical users.
Compatibility	Works smoothly on major browsers and devices (Windows, macOS, Linux).
Maintainability	Modular code structure to allow smooth updates and feature expansion.

5.4 SYSTEM PERFORMANCE GOALS

System performance goals for an AI-driven sign language detection system focus on achieving high accuracy, real-time responsiveness, and robustness across diverse environmental conditions. The system aims to consistently recognize a wide range of signs with precision and recall rates above 95%, ensuring dependable translation for users. Real-time processing speed is targeted to be at least 30 frames per second on standard consumer-grade devices to provide smooth, low-latency interactions. Additionally, the system should maintain stable performance despite variations in hand size, skin tone, lighting, and background complexity. Scalability, ease of integration with common hardware (such as webcams), and user-friendly interaction are also critical goals to maximize accessibility and practical usability in real-world applications

CHAPTER 6
SOFTWARE DESIGN

6.1 USE CASE DIAGRAM

A use case diagram is a visual representation that shows how users interact with a system. It describes the different functions or services the system provides and the relationships between users and these functions. It helps in understanding system requirements and user expectations. Use case diagrams are mainly used during the early stages of software development to plan and communicate system behaviour.

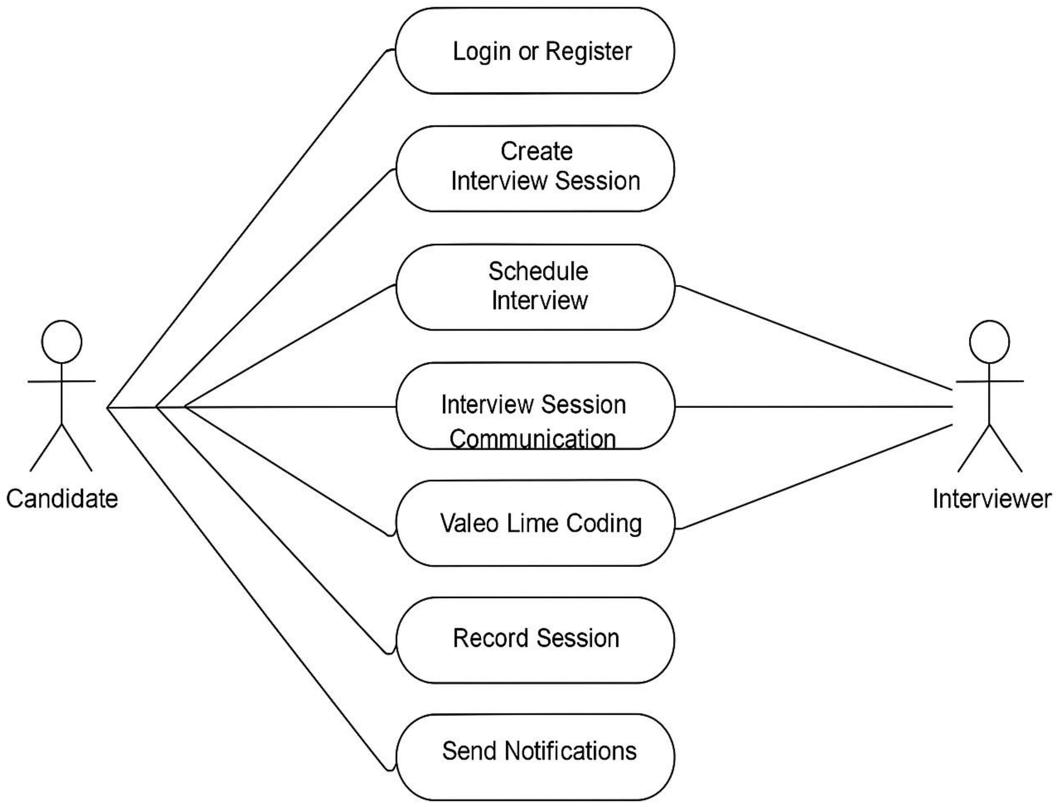


Figure 6.1: Use Case Diagram

6.2 SYSTEM ARCHITECTURE

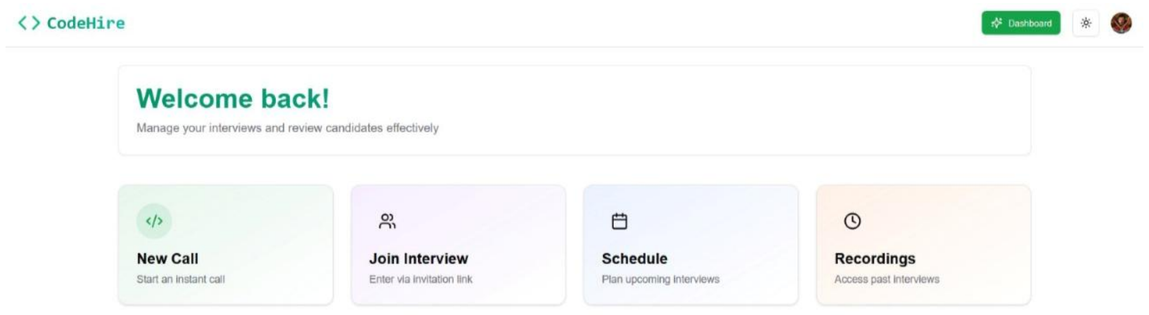
The system architecture for the AI-powered Intelligent Interview and Evaluation Platform Technical Interview Platform follows a structured multi-layered design to ensure efficient performance, real-time communication, and secure data handling. It includes a User Interface (UI) layer, built using React and Next.js, which enables interaction between interviewers and candidates through features such as session dashboards, coding interfaces, and video call screens.

An Application Layer handles essential business logic, including user authentication through Clerk, session creation and management, real-time collaborative coding using the Monaco Editor, and video call operations powered by the Stream Video SDK. This layer also oversees screen sharing, recording, and synchronization of interview data during live sessions.

A dedicated Data Layer, supported by Convex Database, manages secure and real-time storage of user profiles, interview details, coding activity, and session history. An API Layer facilitates smooth communication between the frontend and backend, ensuring fast and consistent data flow across all modules.

The system integrates third-party services such as Stream for real-time communication and Svix for sending notifications and webhook events. Security is maintained through encrypted communication, role-based access control, and protected routes. Cloud-based deployment ensures scalability, reliability, and high availability, allowing organizations and candidates to access interviews anytime with uninterrupted performance.

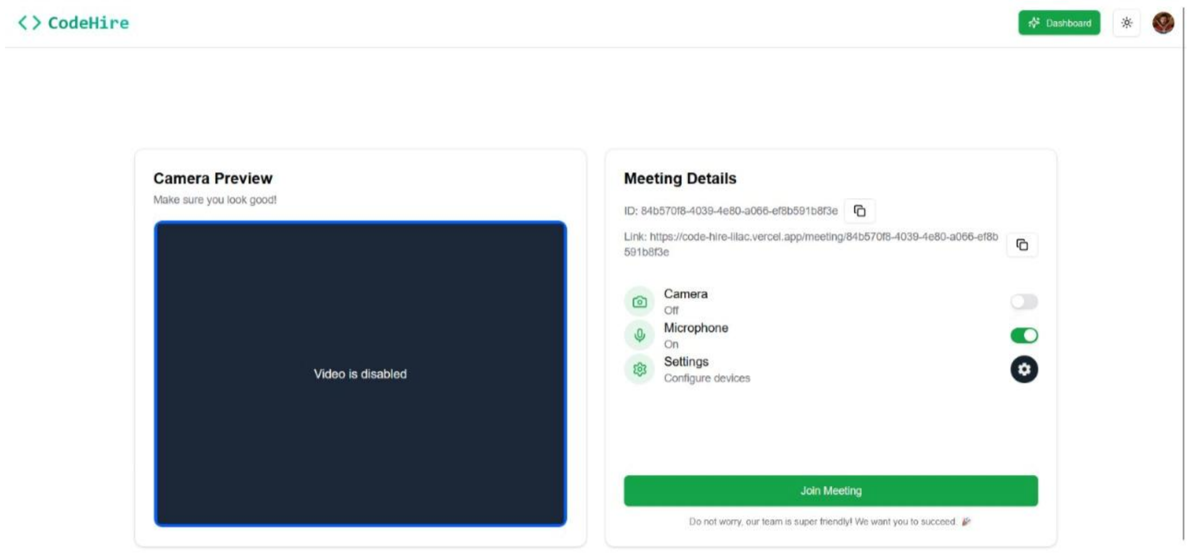
6.3 UI/UXDESIGN



Welcome back!

Manage your interviews and review candidates effectively

Figure 6.2: UI Design Interviewer Dashboard



ID: 84b570f8-4039-4e80-a066-ef8b591b8f3e

Link: https://code-hire-lilac.vercel.app/meeting/84b570f8-4039-4e80-a066-ef8b591b8f3e

Do not worry, our team is super friendly! We want you to succeed. 🎉

Fig 6.3:UI Design Join Meeting Page

CHAPTER 7
WEB MODULES

7.1 Introduction

The **AI-powered Intelligent Interview and Evaluation Platform** platform is built around several core modules that together create a smooth and efficient technical interview experience for users. These modules handle authentication, real-time communication, collaborative coding, session management, and data handling to ensure seamless interaction between interviewers and candidates.

Each module is carefully designed to enhance usability, scalability, and system performance, following the principles of modular architecture. By dividing the system into independent yet interconnected components, the platform achieves improved maintainability, streamlined workflows, and easier integration of future enhancements such as advanced analytics or extended evaluation tools. This modular approach ensures that AI-powered Intelligent Interview and Evaluation Platform remains flexible and adaptable to evolving interview processes and organizational requirements.

7.2 Problem Definition and Module Overview

The primary problem addressed by AI-powered Intelligent Interview and Evaluation Platform is the absence of a unified and structured online environment for conducting technical interviews. Currently, organizations rely on multiple disconnected tools for communication, coding, and evaluation, leading to inefficiencies, interruptions, and inconsistent assessments. To solve this, the system has been divided into several core functional modules, each responsible for a specific area of operation.

7.2.1 User Authentication & Role Management Module - Handles secure login, registration, and role-based authorization for interviewers, candidates, and administrators. It ensures data privacy and controlled access using Clerk-based authentication and protected routes.

7.2.2 Interviewer Dashboard & Session Management Module - Enables interviewers to create interview sessions, schedule meetings, manage candidate participation, and access session recordings. It provides full control over the interview process, ensuring smooth flow and efficient evaluation.

7.2.3 Candidate Interface & Coding Environment Module - Allows candidates to join scheduled sessions, access the Monaco-based coding editor, and collaborate in real time during problem-solving tasks. It integrates syntax highlighting, multi-language support, and code execution for effective technical assessment.

7.2.4 Video Call & Screen Sharing Module - Integrates real-time communication features, enabling 1-to-1 video sessions, live screen sharing, and recorded interactions. This creates a personalized and interactive interview experience while maintaining clarity and transparency.

7.2.5 Admin Management Module - Provides administrative tools to manage platform users, oversee interview activities, verify organization accounts, and maintain system health. It ensures platform security, performance monitoring, and proper governance across all modules.

7.3 Module Description

Each module in the AI-powered Intelligent Interview and Evaluation Platform Platform has been developed using a modern full-stack architecture that ensures efficient workflows, fast performance, and seamless data flow between the frontend, backend, and real-time services.

7.3.1 Frontend (Next.js & React.js)

Implements user interfaces for interviewer dashboards, candidate coding screens, session controls, and video call panels. It provides responsive layouts, smooth navigation, and interactive components for an intuitive and user-friendly experience during technical interviews.

7.3.2 Backend & Real-Time Logic (Convex Server Functions)

Manages core business logic, including session creation, user data handling, interview synchronization, and secure communication between client components. Convex ensures real-time updates across all users without manual API management, enabling smooth collaboration and live coding functionality.

7.3.3 Communication Layer (Stream Video SDK

Handles the video calling, screen sharing, and session recording features for interviewer–candidate interaction. It ensures low-latency, high-quality communication required for effective technical assessments.

7.3.4 Authentication Layer (Clerk)

Provides secure user registration, login, role management, and session control. It ensures that only authorized users can access interview sessions, dashboards, or administrative features.

7.3.5 Database Layer (Convex Storage)

Stores all user profiles, interview sessions, coding activity logs, and system configurations in a structured and scalable format with built-in real-time data access.

7.4 Result Analysis

After integrating all core modules, the **AI-powered Intelligent Interview and Evaluation Platform** Platform demonstrated strong functional and performance outcomes. The authentication and role management system securely handled user access using Clerk, ensuring proper authorization for interviewers, candidates, and administrators. The interviewer session management module functioned effectively, allowing interviewers to create sessions, manage schedules, and review session details without interruptions.

The candidate interface and coding environment operated smoothly, enabling real-time problem solving within the Monaco Editor while maintaining accurate logs of coding activity. The integrated video calling and screen-sharing module offered seamless real-time communication through Stream SDK, allowing interactive and collaborative interview experiences.

CHAPTER 8 CODING

interviews.jsx

```
import StreamClientProvider from "@/components/providers/StreamClientProvider";
function Layout({ children }: { children: React.ReactNode }) {
  return <StreamClientProvider>{children}</StreamClientProvider>;
}
export default Layout;
import { mutation, query } from "./_generated/server";
import { v } from "convex/values";
export const getAllInterviews = query({
  handler: async (ctx) => {
    const identity = await ctx.auth.getUserIdentity();
    if (!identity) throw new Error("Unauthorized");
    const interviews = await ctx.db.query("interviews").collect();
    return interviews;
  },
});
export const getInterviewsForInterviewer = query({
  handler: async (ctx) => {
    const identity = await ctx.auth.getUserIdentity();
    if (!identity) throw new Error("Unauthorized");
    const all = await ctx.db.query("interviews").collect();
    // Filter server-side to only interviews that include this interviewer
    return all.filter((iv) => iv.interviewerIds?.includes(identity.subject));
  },
});
export const getMyInterviews = query({
  handler: async (ctx) => {
    const identity = await ctx.auth.getUserIdentity();
 if (!identity) return [];
```

```
    const interviews = await ctx.db
      .query("interviews")
      .withIndex("by_candidate_id", (q) => q.eq("candidateId", identity.subject))
      .collect();
    return interviews!;
  },
});
export const getInterviewByStreamCallId = query({
  args: { streamCallId: v.string() },
  handler: async (ctx, args) => {
    return await ctx.db
      .query("interviews")
      .withIndex("by_stream_call_id", (q) => q.eq("streamCallId", args.streamCallId))
      .first();
  },
});
export const createInterview = mutation({
  args: {
    title: v.string(),
    description: v.optional(v.string()),
    startTime: v.number(),
    status: v.string(),
    streamCallId: v.string(),
    candidateId: v.string(),
    interviewerIds: v.array(v.string()),
  },
  handler: async (ctx, args) => {
    const identity = await ctx.auth.getUserIdentity();
    if (!identity) throw new Error("Unauthorized");
    return await ctx.db.insert("interviews", {
      ...args,    });  }};
```

Meeting.tsx

```
"use client";
import LoaderUI from "@/components/LoaderUI";
import MeetingRoom from "@/components/MeetingRoom";
import MeetingSetup from "@/components/MeetingSetup";
import useGetCallById from "@/hooks/useGetCallById";
import { useUser } from "@clerk/nextjs";
import { StreamCall, StreamTheme } from "@stream-io/video-react-sdk";
import { useParams } from "next/navigation";
import { useState } from "react";
function MeetingPage() {
  const { id } = useParams();
  const { isLoaded } = useUser();
  const { call, isCallLoading } = useGetCallById(id);
  const [isSetupComplete, setIsSetupComplete] = useState(false);
  if (!isLoaded || isCallLoading) return <LoaderUI />
  if (!call) {
    return (
      <div className="h-screen flex items-center justify-center">
        <p className="text-2xl font-semibold">Meeting not found</p>
      </div>
    );  }
  return (  <StreamCall call={call}>
      <StreamTheme>
        {!isSetupComplete ? (
          <MeetingSetup onSetupComplete={() => setIsSetupComplete(true)} />
        ) : (    <MeetingRoom />
        )}
      </StreamTheme>
    </StreamCall>  );}export default MeetingPage;
```

Dashboard.tsx

```
"use client";
import ActionCard from "@/components/ActionCard";
import { QUICK_ACTIONS } from "@/constants";
import { useUserRole } from "@/hooks/useUserRole";
import { useQuery } from "convex/react";
import { useState } from "react";
import { api } from "../../../../convex/_generated/api";
import { useRouter } from "next/navigation";
import MeetingModal from "@/components/MeetingModal";
import LoaderUI from "@/components/LoaderUI";
import { Loader2Icon } from "lucide-react";
import MeetingCard from "@/components/MeetingCard";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import useMeetingActions from "@/hooks/useMeetingActions";
export default function Home() {
  const router = useRouter();
  const { isInterviewer, isCandidate, isLoading } = useUserRole();
  const interviews = useQuery(api.interviews.getMyInterviews);
  const [showModal, setShowModal] = useState(false);
  const [modalType, setModalType] = useState<"start" | "join">();
  const [joinValue, setJoinValue] = useState("");
  const { joinMeeting } = useMeetingActions();
  const handleQuickAction = (title: string) => {
    switch (title) {
      case "New Call":
        setModalType("start");
        setShowModal(true);
        break;
      case "Join Interview":
```

```
 setModalType("join");
        setShowModal(true);

        break;
      default:
        router.push(`/${title.toLowerCase()}`);
    }};
  if (isLoading) return <LoaderUI />;
  return (
    <div className="container max-w-7xl mx-auto p-6">
      {/* WELCOME SECTION */}
      <div className="rounded-lg bg-card p-6 border shadow-sm mb-10">
        <h1 className="text-4xl font-bold bg-gradient-to-r from-emerald-600 to-teal-500 bg-clip-text
text-transparent">
          Welcome back!
        </h1>
        <p className="text-muted-foreground mt-2">
          {isInterviewer
            ? "Manage your interviews and review candidates effectively"
            : "Access your upcoming interviews and preparations"}
        </p>
        {/* JOIN BY ID - candidates only */}
        {isCandidate && (
          <div className="mt-6 grid gap-3 sm:flex sm:items-center">
            <Input
              placeholder="Enter meeting ID or paste meeting link..."
              value={joinValue}
              onChange={(e) => setJoinValue(e.target.value)}
              className="sm:max-w-md"
            />
            <Button
```

```
onClick={() => {
        const raw = joinValue.trim();
        if (!raw) return;
        let meetingId = raw;
        try {

         const url = new URL(raw);
         const parts = url.pathname.split("/").filter(Boolean);
         meetingId = parts[parts.length - 1] || raw;
        } catch {
         const parts = raw.split("/").filter(Boolean);
         meetingId = parts[parts.length - 1] || raw;
        }
       joinMeeting(meetingId);
       }}
       disabled={!joinValue.trim()}
       Join Interview
      </Button>
     </div>
    )}
   </div>
   {isInterviewer ? (<>
     <div className="grid sm:grid-cols-2 lg:grid-cols-4 gap-6">
      {QUICK_ACTIONS.map((action) => (
       <ActionCard
        key={action.title}
        action={action}
        onClick={() => handleQuickAction(action.title)} />
      ))}
     </div>
     <MeetingModal
```

```
isOpen={showModal}
    onClose={() => setShowModal(false)}
    title={modalType === "join" ? "Join Meeting" : "Start Meeting"}
    isJoinMeeting={modalType === "join"}
  />
 </>
) : (
 <>  <div>

    <h1 className="text-3xl font-bold">Your Interviews</h1>
    <p className="text-muted-foreground mt-1">View and join your scheduled interviews</p>
  </div>
  <div className="mt-8">
    {interviews === undefined ? (
     <div className="flex justify-center py-12">
      <Loader2Icon className="h-8 w-8 animate-spin text-muted-foreground" />
     </div>
    ) : interviews.length > 0 ? (
     <div className="grid gap-6 md:grid-cols-2 lg:grid-cols-3">
      {interviews.map((interview) => (
       <MeetingCard key={interview._id} interview={interview} />
      ))}
     </div>
    ) : (
     <div className="text-center py-12 text-muted-foreground">
      You have no scheduled interviews at the moment
     </div>
    )}   </div>   </>)}  </div>  );}
```

Recordings.tsx

```
"use client";
import LoaderUI from "@/components/LoaderUI";
import RecordingCard from "@/components/RecordingCard";
import { ScrollArea } from "@/components/ui/scroll-area";
import useGetCalls from "@/hooks/useGetCalls";
import { CallRecording } from "@stream-io/video-react-sdk";
import { useEffect, useState } from "react";
function RecordingsPage() {
  const { calls, isLoading } = useGetCalls();
  const [recordings, setRecordings] = useState<CallRecording[]>([]);
  useEffect(() => {
    const fetchRecordings = async () => {

      if (!calls) return;
      try {
        // Get recordings for each call
        const callData = await Promise.all(calls.map((call) => call.queryRecordings()));
        const allRecordings = callData.flatMap((call) => call.recordings);

        setRecordings(allRecordings);
      } catch (error) {
        console.log("Error fetching recordings:", error);
      }};
    fetchRecordings();
  }, [calls]);
  if (isLoading) return <LoaderUI />;
  return (
    <div className="container max-w-7xl mx-auto p-6">
      {/* HEADER SECTION */}
      <h1 className="text-3xl font-bold">Recordings</h1>
      <p className="text-muted-foreground my-1">
        {recordings.length} {recordings.length === 1 ? "recording" : "recordings"} available
      </p>
      {/* RECORDINGS GRID */}
      <ScrollArea className="h-[calc(100vh-12rem)] mt-3">
        {recordings.length > 0 ? (
          <div className="grid grid-cols-1 md:grid-cols-2 xl:grid-cols-3 gap-6 pb-6">
            {recordings.map((r) => (
              <RecordingCard key={r.end_time} recording={r} />
            ))}
          </div>
        ) : (
          <div className="flex flex-col items-center justify-center h-[400px] gap-4">
            <p className="text-xl font-medium text-muted-foreground">No recordings available</p>
          </div>
        )}

      </ScrollArea>
    </div>
  );
}
export default RecordingsPage;
```

InterviewSchedule.tsx

```
import { useUser } from "@clerk/nextjs";
import { useStreamVideoClient } from "@stream-io/video-react-sdk";
import { useMutation, useQuery } from "convex/react";
import { useState } from "react";
import { api } from "../../../../convex/_generated/api";
import toast from "react-hot-toast";
import {
  Dialog,
  DialogHeader,
  DialogTitle,
  DialogTrigger,
  DialogContent,
} from "@/components/ui/dialog";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Textarea } from "@/components/ui/textarea";
import {
  Select,
  SelectContent,
  SelectItem,
  SelectTrigger,
  SelectValue,
} from "@/components/ui/select";
import UserInfo from "@/components/UserInfo";


import { Loader2Icon, XIcon } from "lucide-react";

import { Calendar } from "@/components/ui/calendar";
import { TIME_SLOTS } from "@/constants";
import MeetingCard from "@/components/MeetingCard";
function InterviewScheduleUI() {
  const client = useStreamVideoClient();
  const { user } = useUser();
  const [open, setOpen] = useState(false);
  const [isCreating, setIsCreating] = useState(false);

  const interviews = useQuery(api.interviews.getInterviewsForInterviewer) ?? [];
  const users = useQuery(api.users.getUsers) ?? [];
  const createInterview = useMutation(api.interviews.createInterview);
  const deleteInterview = useMutation(api.interviews.deleteInterview);

  const candidates = users?.filter((u) => u.role === "candidate");
  const interviewers = users?.filter((u) => u.role === "interviewer");

  const [formData, setFormData] = useState({
    title: "",
    description: "",
    date: new Date(),
    time: "09:00",
    candidateId: "",
    interviewerIds: user?.id ? [user.id] : [],
  });

  const scheduleMeeting = async () => {
    if (!client || !user) return;
    if (!formData.candidateId || formData.interviewerIds.length === 0) {
      toast.error("Please select both candidate and at least one interviewer");
      return;
    }
    setIsCreating(true);
```

```
try {
  const { title, description, date, time, candidateId, interviewerIds } = formData;
  const [hours, minutes] = time.split(":");
  const meetingDate = new Date(date);
  meetingDate.setHours(parseInt(hours), parseInt(minutes), 0);

  const id = crypto.randomUUID();
  const call = client.call("default", id);
  await call.getOrCreate({
    data: {
      starts_at: meetingDate.toISOString(),
      custom: {
        description: title,
        additionalDetails: description,
      },
    },
  });
  await createInterview({
    title,
    description,
    startTime: meetingDate.getTime(),
    status: "upcoming",
    streamCallId: id,
    candidateId,
    interviewerIds,
  });
  setOpen(false);
  toast.success("Meeting scheduled successfully!");
  setFormData({
    title: "",
    description: "",
      date: new Date(),
       time: "09:00",
       candidateId: "",
       interviewerIds: user?.id ? [user.id] : [],
     });
   } catch (error) {
     console.error(error);
     toast.error("Failed to schedule meeting. Please try again.");
   } finally {
     setIsCreating(false);
   }  };
  const addInterviewer = (interviewerId: string) => {
   if (!formData.interviewerIds.includes(interviewerId)) {
     setFormData((prev) => ({
       ...prev,
       interviewerIds: [...prev.interviewerIds, interviewerId],
     }));    }  };
  const removeInterviewer = (interviewerId: string) => {
   if (interviewerId === user?.id) return;
   setFormData((prev) => ({
     ...prev,
     interviewerIds: prev.interviewerIds.filter((id) => id !== interviewerId),
   }));};
  const selectedInterviewers = interviewers.filter((i) =>
   formData.interviewerIds.includes(i.clerkId)
  );
  const availableInterviewers = interviewers.filter(
   (i) => !formData.interviewerIds.includes(i.clerkId)
  );
  return (
   <div className="container max-w-7xl mx-auto p-6 space-y-8">
     <div className="flex items-center justify-between">
```

```
{/* HEADER INFO */}
<div>
  <h1 className="text-3xl font-bold">Interviews</h1>
  <p className="text-muted-foreground mt-1">Schedule and manage interviews</p>
</div>
{/* DIALOG */}
<Dialog open={open} onOpenChange={setOpen}>
  <DialogTrigger asChild>
    <Button size="lg">Schedule Interview</Button>
  </DialogTrigger>
  <DialogContent className="sm:max-w-[500px] h-[calc(100vh-200px)] overflow-auto">
    <DialogHeader>
      <DialogTitle>Schedule Interview</DialogTitle>
    </DialogHeader>
    <div className="space-y-4 py-4">
      {/* INTERVIEW TITLE */}
      <div className="space-y-2">
        <label className="text-sm font-medium">Title</label>
        <Input
          placeholder="Interview title"
          value={formData.title}
          onChange={(e) => setFormData({ ...formData, title: e.target.value })}/>
      </div>
      {/* INTERVIEW DESC */}
      <div className="space-y-2">
        <label className="text-sm font-medium">Description</label>
        <Textarea
          placeholder="Interview description"
          value={formData.description}
          onChange={(e) => setFormData({ ...formData, description: e.target.value })}
          rows={3}/>
      </div>
      {/* CANDIDATE */}
      <div className="space-y-2">
        <label className="text-sm font-medium">Candidate</label>
        <Select
          value={formData.candidateId}
          onValueChange={(candidateId) => setFormData({ ...formData, candidateId })}
        >
          <SelectTrigger>
            <SelectValue placeholder="Select candidate" />
          </SelectTrigger>
          <SelectContent>
            {candidates.map((candidate) => (
              <SelectItem key={candidate.clerkId} value={candidate.clerkId}>
                <UserInfo user={candidate} />
              </SelectItem>
            ))}
          </SelectContent>
        </Select>
      </div>
      {/* INTERVIEWERS */}
      <div className="space-y-2">
        <label className="text-sm font-medium">Interviewers</label>
        <div className="flex flex-wrap gap-2 mb-2">
          {selectedInterviewers.map((interviewer) => (
            <div
              key={interviewer.clerkId}
              className="inline-flex items-center gap-2 bg-secondary px-2 py-1 rounded-md text-
sm">
              <UserInfo user={interviewer} />
              {interviewer.clerkId !== user?.id && (
                <button
                  onClick={() => removeInterviewer(interviewer.clerkId)}
                  className="hover:text-destructive transition-colors">
                  <XIcon className="h-4 w-4" />
                </button>
              )}
            </div>
          ))}
        </div>
        {availableInterviewers.length > 0 && (
          <Select onValueChange={addInterviewer}>
            <SelectTrigger>
              <SelectValue placeholder="Add interviewer" />
            </SelectTrigger>
            <SelectContent>
              {availableInterviewers.map((interviewer) => (
                <SelectItem key={interviewer.clerkId} value={interviewer.clerkId}>
                  <UserInfo user={interviewer} />
                </SelectItem>
              ))}
            </SelectContent>
          </Select>
        )}
      </div>
      {/* DATE & TIME */}
      <div className="flex gap-4">
        {/* CALENDAR */}
        <div className="space-y-2">
          <label className="text-sm font-medium">Date</label>
          <Calendar
            mode="single"
```

```
          selected={formData.date}
          onSelect={(date) => date && setFormData({ ...formData, date })}
          disabled={(date) => date < new Date()}
          className="rounded-md border"   />
      </div>
      {/* TIME */}
      <div className="space-y-2">
        <label className="text-sm font-medium">Time</label>
        <Select
          value={formData.time}
          onValueChange={(time) => setFormData({ ...formData, time })}>
          <SelectTrigger>
            <SelectValue placeholder="Select time" />
          </SelectTrigger>
          <SelectContent>
            {TIME_SLOTS.map((time) => (
              <SelectItem key={time} value={time}>
                {time}
              </SelectItem>
            ))}
          </SelectContent>
        </Select>
      </div>
    </div>
    <div className="flex justify-end gap-3 pt-4">
      <Button variant="outline" onClick={() => setOpen(false)}>
        Cancel</Button>
      <Button onClick={scheduleMeeting} disabled={isCreating}>
        {isCreating ? (
          <><Loader2Icon className="mr-2 size-4 animate-spin" />
            Scheduling...
```

```
                </>) : (
                "Schedule Interview")}
              </Button>
            </div>
          </div>
        </DialogContent>
      </Dialog>
    </div>
    {/* LOADING STATE & MEETING CARDS */}
    {!interviews ? (
      <div className="flex justify-center py-12">
        <Loader2Icon className="size-8 animate-spin text-muted-foreground" />
      </div>
    ) : interviews.length > 0 ? (
      <div className="spacey-4">
        <div className="grid gap-6 md:grid-cols-2 lg:grid-cols-3">
          {interviews.map((interview) => (
            <MeetingCard
              key={interview._id}
              interview={interview}
              onDelete={async (id) => {
                try {
                  await deleteInterview({ id });
                  toast.success("Interview deleted");
                } catch (e) {
                  console.error(e);
                  toast.error("Failed to delete interview");
                }  }}   />
          ))}
export default InterviewScheduleUI;
```

ChooseRole.tsx

```
"use client";
import { useUser } from "@clerk/nextjs";
import { useMutation } from "convex/react";
import { api } from "../../../convex/_generated/api";
import { useRouter } from "next/navigation";
import { useState } from "react";
import { Button } from "@/components/ui/button";
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from
"@/components/ui/card";
import { UserIcon, BriefcaseIcon } from "lucide-react";

export default function ChooseRolePage() {
  const { user, isLoaded } = useUser();
  const router = useRouter();
  const updateRole = useMutation(api.users.updateUserRole);
  const [submitting, setSubmitting] = useState<"candidate" | "interviewer" | null>(null);

  if (!isLoaded) return null;

  const onChoose = async (role: "candidate" | "interviewer") => {
    if (!user) return;
    try {
      setSubmitting(role);
      await updateRole({ clerkId: user.id, role });
      // Redirect to home (or dashboard) after setting
      router.replace("/");
    } catch (e) {
      console.error(e);
      setSubmitting(null);
      alert("Failed to set role. Please try again.");
    }
  };

  return (
    <div className="min-h-[70vh] flex items-center justify-center">
      <Card className="w-full max-w-xl">
        <CardHeader>
          <CardTitle>Choose your role</CardTitle>
          <CardDescription>
            Select how you want to use AI-powered Intelligent Interview and Evaluation Platform. You
can switch later from your profile if needed.
          </CardDescription>
        </CardHeader>
        <CardContent>
          <div className="grid grid-cols-1 sm:grid-cols-2 gap-4">
            <div className="border rounded-lg p-6 flex flex-col items-center text-center gap-3">
              <UserIcon className="h-10 w-10 text-emerald-500" />
              <h3 className="font-semibold">Candidate</h3>
              <p className="text-sm text-muted-foreground">
                Practice problems, join interviews, and review feedback.
              </p>
              <Button className="mt-2 w-full" onClick={() => onChoose("candidate")}
disabled={submitting !== null}>
                {submitting === "candidate" ? "Selecting..." : "Continue as Candidate"}
              </Button>
            </div>

            <div className="border rounded-lg p-6 flex flex-col items-center text-center gap-3">
              <BriefcaseIcon className="h-10 w-10 text-teal-500" />
              <h3 className="font-semibold">Interviewer</h3>
              <p className="text-sm text-muted-foreground">
```

```
                Schedule interviews, manage candidates, and leave evaluations.
              </p>
              <Button variant="secondary" className="mt-2 w-full" onClick={() =>
onChoose("interviewer")} disabled={submitting !== null}>
                {submitting === "interviewer" ? "Selecting..." : "Continue as Interviewer"}
              </Button>
            </div>
          </div>
        </CardContent>
      </Card>
    </div>
  );
}
```

Globals.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

body {
  font-family: Arial, Helvetica, sans-serif;
}

@layer utilities {
  .text-balance {
    text-wrap: balance;
  }
}

@layer base {
  :root {
    --background: 0 0% 100%;
  --foreground: 240 10% 3.9%;
    --card: 0 0% 100%;
    --card-foreground: 240 10% 3.9%;
    --popover: 0 0% 100%;
    --popover-foreground: 240 10% 3.9%;
    --primary: 142.1 76.2% 36.3%;
    --primary-foreground: 355.7 100% 97.3%;
    --secondary: 240 4.8% 95.9%;
    --secondary-foreground: 240 5.9% 10%;
    --muted: 240 4.8% 95.9%;
    --muted-foreground: 240 3.8% 46.1%;
    --accent: 240 4.8% 95.9%;
    --accent-foreground: 240 5.9% 10%;
    --destructive: 0 84.2% 60.2%;
    --destructive-foreground: 0 0% 98%;
    --border: 240 5.9% 90%;
    --input: 240 5.9% 90%;
    --ring: 142.1 76.2% 36.3%;
    --radius: 0.5rem;
    --chart-1: 12 76% 61%;
    --chart-2: 173 58% 39%;
    --chart-3: 197 37% 24%;
    --chart-4: 43 74% 66%;
    --chart-5: 27 87% 67%;
  }

  .dark {
    --background: 20 14.3% 4.1%;
    --foreground: 0 0% 95%;
    --card: 24 9.8% 10%;


    --card-foreground: 0 0% 95%;
    --popover: 0 0% 9%;
    --popover-foreground: 0 0% 95%;
    --primary: 142.1 70.6% 45.3%;
    --primary-foreground: 144.9 80.4% 10%;
    --secondary: 240 3.7% 15.9%;
    --secondary-foreground: 0 0% 98%;
    --muted: 0 0% 15%;
    --muted-foreground: 240 5% 64.9%;
    --accent: 12 6.5% 15.1%;
    --accent-foreground: 0 0% 98%;
    --destructive: 0 62.8% 30.6%;
    --destructive-foreground: 0 85.7% 97.3%;
    --border: 240 3.7% 15.9%;
    --input: 240 3.7% 15.9%;
    --ring: 142.4 71.8% 29.2%;
    --chart-1: 220 70% 50%;
    --chart-2: 160 60% 45%;
    --chart-3: 30 80% 55%;
    --chart-4: 280 65% 60%;
    --chart-5: 340 75% 55%;
  }
}

@layer base {
  * {
    @apply border-border;
  }
  body {
    @apply bg-background text-foreground;}}
```

CHAPTER 9
RESULTS AND OUTPUT SCREEN

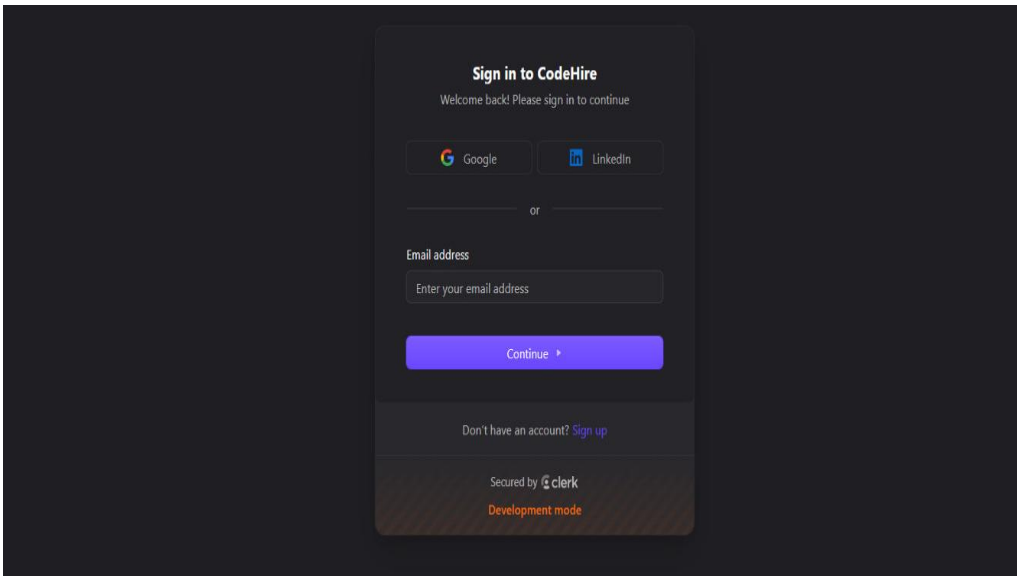


Figure 9.1 Login Page

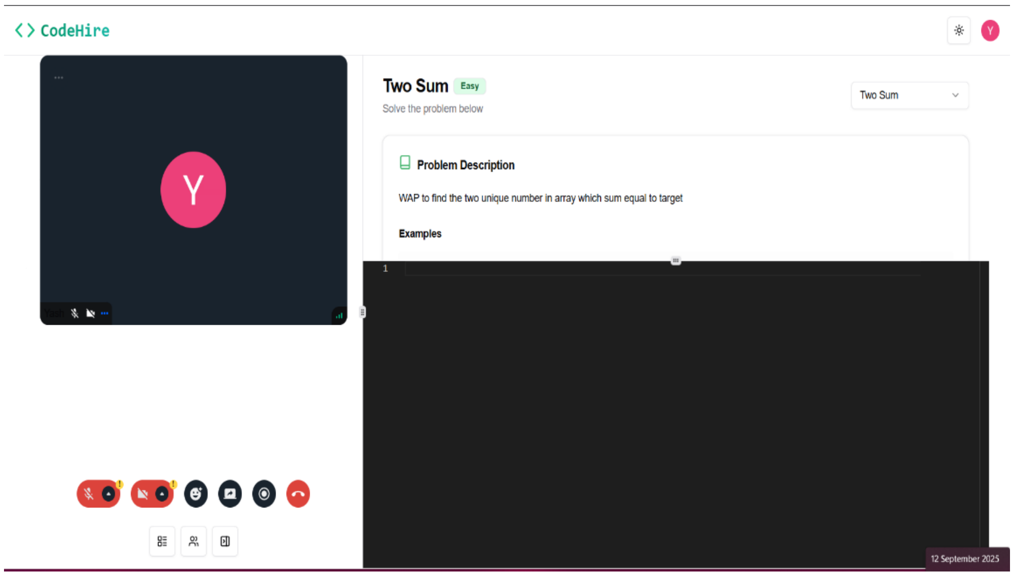
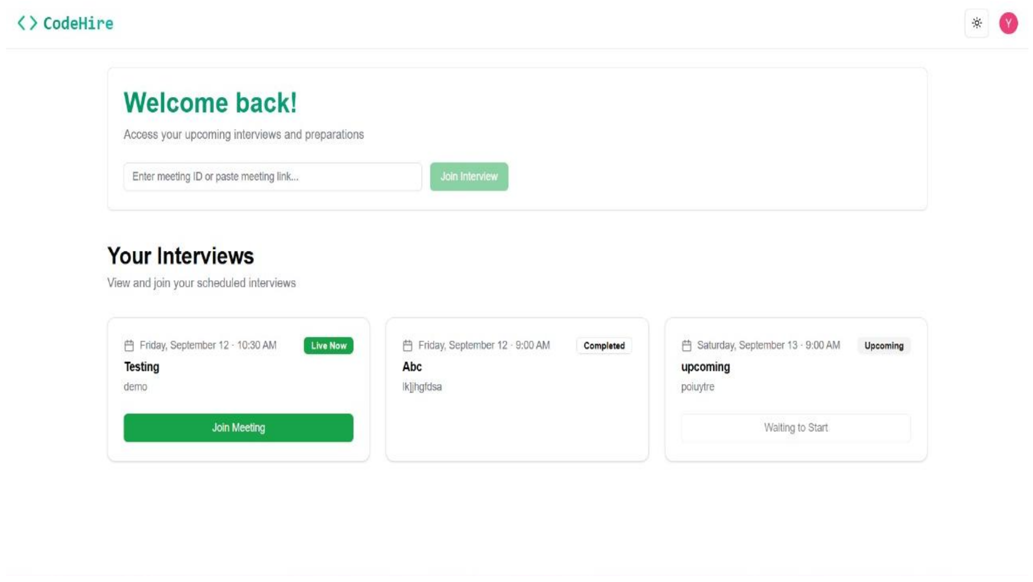


Figure 9.2: Interview Page


<> CodeHire

Welcome back!

Access your upcoming interviews and preparations

Enter meeting ID or paste meeting link...

Join Interview

Your Interviews

View and join your scheduled interviews

Friday, September 12 · 10:30 AM Live Now

Testing

demo

Join Meeting

Friday, September 12 · 9:00 AM Completed

Abc

lkjhgfdsa

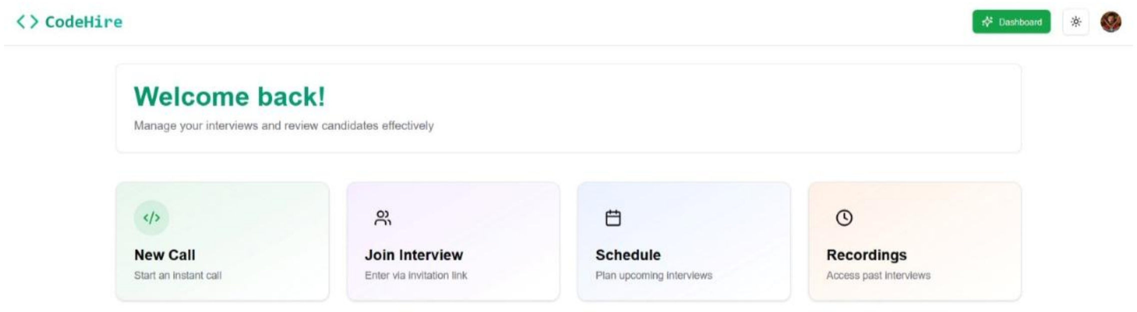
Saturday, September 13 · 9:00 AM Upcoming

upcoming

poiuytre

Waiting to Start

Figure 9.3: Candidate Dashboard


<> CodeHire

Dashboard

Welcome back!

Manage your interviews and review candidates effectively

New Call

Start an instant call

Join Interview

Enter via invitation link

Schedule

Plan upcoming interviews

Recordings

Access past interviews

Figure 9.4: Interviewer Dashboard

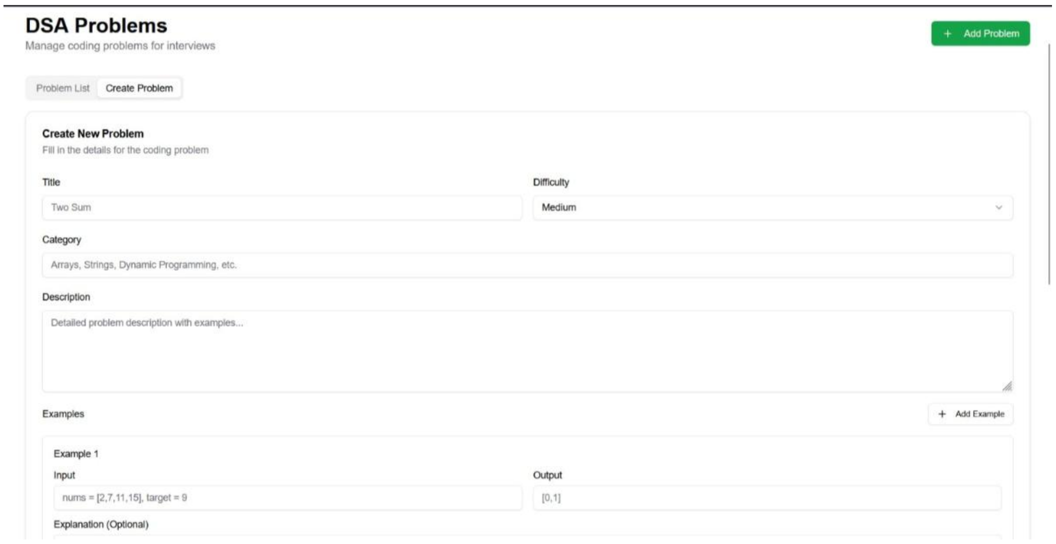

DSA Problems
Manage coding problems for interviews
+ Add Problem
Problem List | Create Problem
Create New Problem
Fill in the details for the coding problem
Title: Two Sum
Difficulty: Medium
Category: Arrays, Strings, Dynamic Programming, etc.
Description: Detailed problem description with examples...
Examples
+ Add Example
Example 1
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation (Optional)

Figure 9.5: Coding Problem Editor Page

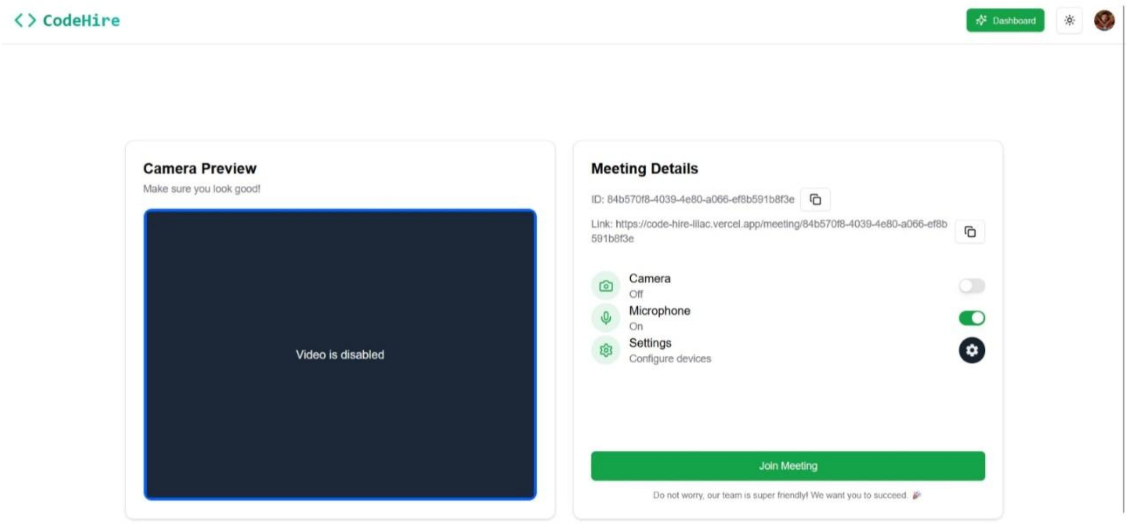

<> CodeHire
Dashboard
Camera Preview
Make sure you look good!
Video is disabled
Meeting Details
ID: 84b570f8-4039-4e80-a066-ef8b591b8f3e
Link: https://code-hire-lilac.vercel.app/meeting/84b570f8-4039-4e80-a066-ef8b591b8f3e
Camera: Off
Microphone: On
Settings: Configure devices
Join Meeting
Do not worry, our team is super friendly! We want you to succeed. 🎉

Figure 9.6: Meeting Page

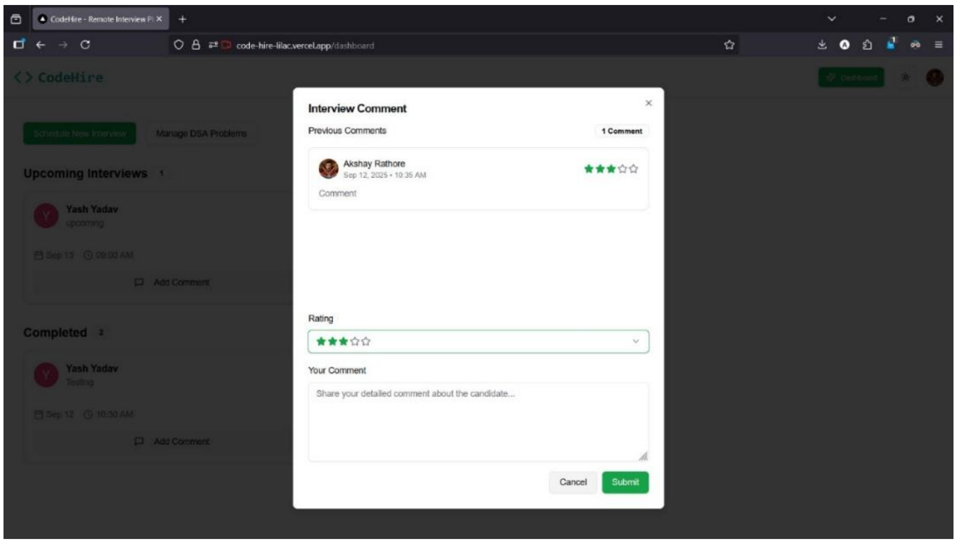


Figure 9.7: Feedback Page

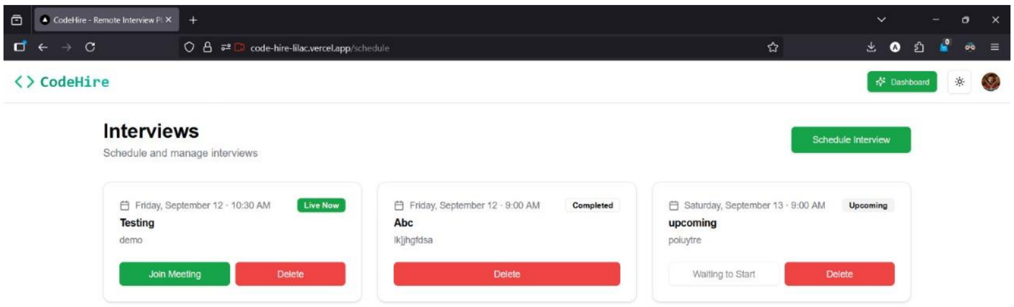

CodeHire - Remote Interview Pl
code-hire-lilac.vercel.app/schedule
CodeHire
Dashboard
Interviews
Schedule and manage interviews
Schedule Interview
Friday, September 12 · 10:30 AM
Live Now
Testing
demo
Join Meeting
Delete
Friday, September 12 · 9:00 AM
Completed
Abc
lkjhgfdsa
Delete
Saturday, September 13 · 9:00 AM
Upcoming
upcoming
poiuytre
Waiting to Start
Delete

Figure 9.8: Interview Scheduler Page

CHAPTER 10
CONCLUSION AND FUTURE WORKS

10.1 Conclusion

In conclusion, the AI-powered Intelligent Interview and Evaluation Platform Technical Interview Platform is a comprehensive solution designed to streamline and modernize the process of conducting technical assessments. The platform emphasizes ease of use, real-time collaboration, and structured interview workflows, ensuring that candidates can showcase their skills effectively while interviewers can evaluate them through a unified and professional system. By integrating core modules such as real-time video calling, screen sharing, a collaborative Monaco-based coding editor, and session management tools, the platform provides a seamless and engaging interview experience.
As AI-powered Intelligent Interview and Evaluation Platform continues to grow, it holds strong potential to make technical hiring more accessible, efficient, and transparent for organizations of all sizes. The structured environment supports fair evaluation, reduces dependency on multiple disconnected tools, and enhances the overall quality of interview interactions. With future enhancements such as expanded analytics, automated session summaries, and scalable premium features, the system aims to ensure long-term sustainability while delivering meaningful value to both interviewers and candidates, creating a reliable digital interview ecosystem.

10.2 Future Enhancements

To further enhance the AI-powered Intelligent Interview and Evaluation Platform Platform, the following improvements can be considered:

Introduce detailed analytics dashboards to provide insights into interviewer performance, candidate engagement, and session outcomes.

Implement automated session summaries and structured feedback reports to streamline post-interview evaluation.

Offer essential interview features for free while providing premium tools such as extended recording access, additional collaboration features, and organizational dashboards to support sustainability.

REFERENCES

JOURNALS / RESEARCH PAPERS

[1]. Author surname, initials. (Year) Title of article, Journal name, Volume number (Issue or part number), first and last page numbers.

[2]. Chakraborty, S., and Saha, P. (2022) 'Enhancing Remote Coding Interviews Using Real-Time Collaboration Tools', *International Journal of Computer Applications*, Vol.175(5), pp.34–40.

[3]. Sharma, R., and Mehta, A. (2021) 'Integrating Cloud-Based IDEs for Technical Skill Assessment', *IEEE Access*, Vol.9, pp.12563–12572.

[4]. Liu, H., and Wang, J. (2020) 'A Study on WebRTC-Based Real-Time Communication Systems', *Journal of Web Engineering*, Vol.18(3), pp.411–428.

BOOKS

[5]. Pressman, R.S., and Maxim, B.R., *Software Engineering: A Practitioner's Approach*, McGraw-Hill Education, 9th Edition, 2019.

[6]. Flanagan, D., *JavaScript: The Definitive Guide*, O'Reilly Media, 7th Edition, 2020.

[7]. Duckett, J., *HTML & CSS: Design and Build Websites*, Wiley, 1st Edition, 2011.

[8]. Snyder, C., *Designing for the Digital Age: How to Create Human-Centered Products and Services*, New Riders, 1st Edition, 2007.

WEBSITES

[9]. https://react.dev/learn

[10]. https://nodejs.org/en/docs

[11]. https://nextjs.org/docs

[12]. https://www.stream.io/docs/video

PROJECT SUMMARY

About Project

Title of the project	AI-POWERED INTELLIGENT INTERVIEW AND EVALUATION PLATFORM
Semester	7th Semester
Members	Yash Yadav Raj Vijsy Mahajan
Team Leader	Raj Mahajan
Describe role of every member in the project	**Yash Yadav:** Backend + Connectivity **Raj Vijay Mahajan:** Frontend + UI/UX
What is the motivation for selecting this project?	This project was selected to help interviewers and candidates conduct real technical interviews through a unified and interactive digital platform.
Project Type (Desktop Application, Web Application, Mobile App, Web)	Web Application

Tools & Technologies

Programming language used	JavaScript, Node.js, Express.js, React.js, TypeScript
Compiler used (with version)	Browser Rendering Engines (Chrome V8)
IDE used (with version)	Visual Studio Code (Version1.51.1)
Front End Technologies (with version, wherever Applicable)	Next.js (Version 14) React.js (Version 19) Tailwind CSS
Back End Technologies (with version, wherever applicable)	Convex Backend (Latest Stable Release) Stream Video SDK (Latest Version) Clerk Authentication SDK (Latest Version) Svix Webhooks (Latest Version)
Database used (with version)	Convex Database (Cloud-hosted)

<u>Software Design & Coding</u>

SDLC model followed (Waterfall, Agile, Spiral etc.)	Agile Model
Why is the above SDLC model followed?	Agile was used because it allows fast updates, continuous improvements, and easy modification of real-time features like video calling and coding collaboration based on testing and user feedback.
Justify that the SDLC model mentioned above is followed in the project.	Agile was followed because development was done in small iterations with continuous testing and improvements.
Software Design approach followed (Functional or Object Oriented)	Object-Oriented Design
Name the diagrams developed (According to the Design approach followed)	Use Case Diagram
In case Object Oriented approach is followed, which of the OOPS principles are covered in design?	Encapsulation, Abstraction, Inheritance, Polymorphism
No. of Tiers(example3-tier)	3-tier
Total no. of front-end pages	8 front-end pages
Total no. of collections in database	4 collections
Database in which Normal Form?	3rd Normal Form (3NF)
Are the entries in the database encrypted?	Yes
Frontend validations applied (Yes /No)	Yes
Session management done (in case of web applications)	Yes (Token-based)
Exception handling done	Yes
Commenting done in code (Yes /No)	Yes
Naming convention followed (Yes /No)	Yes

What difficulties faced during deployment of the project?	Deployment faced minor issues with real-time service integration, environment variable configuration (Stream & Clerk), and version conflicts during Next.js build.
Total no. of Use-cases	5
Give titles of Use-cases	Login & Authentication Create / Join Interview Session Live Coding Collaboration View Session Summary / Recording Admin Controls & User Management

Project Requirements

MVC architecture followed (Yes /No)	Yes
If yes, write the name of MVC architecture followed (MVC-1, MVC-2)	MVC-2
Design Pattern used (Yes /No)	Yes
If yes, write the name of Design Pattern used	Observer Pattern (for real-time updates)
Interface type (CLI /GUI)	Graphical User Interface (GUI)
No. of Actors	2
Name of Actors	User (Interviewer / Candidate) , Admin
Total no. of Functional Requirements	8 Functional Requirements
List few important non- Functional Requirements	Performance, Security, Scalability, Usability, Reliability

Testing

Which testing is performed? (Manual or Automation)	Manual testing is performed
Is Beta testing done for this project?	Yes

Write project narrative covering above mentioned points

The AI-powered Intelligent Interview and Evaluation Platform Technical Interview Platform was developed using the Agile SDLC model, as it allowed continuous improvement through iterative development, testing, and feedback collection. This approach helped refine real-time features such as video calling, collaborative coding, screen sharing, and session management based on user experience and performance results. The project follows an Object-Oriented Design approach and uses the MVC-2 architecture, ensuring a clear separation between the user interface, application logic, and data handling layers. To support real-time updates during coding sessions and live interviews, the Observer Design Pattern was implemented, enabling synchronized changes between interviewer and candidate screens.

The system provides a GUI-based interface built with Next.js, Tailwind CSS, and Radix UI to ensure a modern and responsive user experience. The platform includes two primary actors: the User (interviewer or candidate) who participates in the interview, and the Admin, who manages system operations, users, and platform activity. The project includes eight core functional requirements, such as authentication, session creation, video calling, collaborative coding, session recording, and administrative controls. Key non-functional requirements like security, performance, scalability, usability, and reliability were prioritized to ensure stable and high-quality interview experiences.

Yash Yadav	0187CS221219
Raj Vijay Mahajan	0187CS221159

Guide Signature

Dr. Amit Dubey

APPENDIX-1 GLOSSARY OF TERMS

(In alphabetical order)

A

API	A set of rules allowing communication between frontend and backend systems.

B

BACKEND	Server-side part of the application responsible for logic, processing, and data handling.

D

DATABASE	Storage system for user data and interview records.

F

FRONT-END	The user-facing part of the platform, built with technologies like HTML, CSS, and JavaScript frameworks.

G

GUI	A visual interface that allows users to interact with the system easily.

I

INTERVIEW PANEL	A set of AI-driven virtual interviewers simulating a boardroom environment.

N

NLP	AI technique used to understand and evaluate human language.

S

SPEECH-TO-TEXT	Technology that converts spoken words into written text.

U

USER AUTHENTICATION	The process of verifying user identity through login credentials.