AI Developer Coding and Skill Assessment

This assessment is divided into two key tasks:

1. Task 1: AI-Powered Interview Application (LiveKit + OpenAI)
2. Task 2: AI-Driven Job Posting, Scheduling & Recruiter Assistant (OpenAI + LangChain + API Integrations)

Task 1: AI-Powered Interview Application (LiveKit + OpenAI)

Goal of the Task

Develop an AI-powered interview application where:

- Candidates upload resumes.
- AI Agent (Avatar) conducts real-time interviews.
- AI dynamically generates questions and evaluates responses.
- AI-driven analytics provide feedback on candidate performance.

This system will leverage LiveKit for real-time video, OpenAI for AI-driven interviews, and HeyGen for an AI avatar.

Architecture Overview

Key Components

1. Frontend (Next.js, WebRTC, LiveKit SDK)
    - UI for candidates to upload resumes.
    - Video streaming integration with LiveKit.
    - Display AI Avatar using HeyGen.
2. Backend (Node.js, Express.js, OpenAI API, LangChain, PostgreSQL)
    - Handles authentication, resume processing, and AI model integration.
    - Manages AI-driven interview logic and stores analytics.
3. AI & Video Streaming (LiveKit + OpenAI + HeyGen)
    - LiveKit manages video rooms and real-time interactions.
    - OpenAI generates dynamic interview questions & feedback.
    - HeyGen provides AI Avatar responses.

Key Functionalities to Achieve

1. Resume Upload and Processing

- Candidates upload resumes in PDF/DOC format.
- AI extracts skills, experience, education (via OpenAI embeddings).

2. Video Room Management (LiveKit)

- Create unique interview rooms.

- Display candidate & AI Avatar streams.
- Manage real-time video/audio interactions.

3. LiveKit Agents for AI Interaction

- AI agent conducts interview, adapting questions dynamically.
- Uses OpenAI GPT-4-turbo for real-time conversation.
- Adjusts responses based on NLP & resume context.

Example AI-driven interactions:

- Question Generation: “Tell me about a time you solved a complex problem.”
- Follow-up Questions: Based on candidate’s response.
- Real-time Sentiment Analysis: Evaluate confidence levels.

4. LangChain for AI Assistants

- Stores conversation context using LangChain Memory.
- Uses LLM Chains to personalize interviews.
- Retrieves relevant industry-specific questions.

5. FAISS & ChromaDB for AI Search

- Vector search for similar past interview responses.
- Ranks candidate responses against benchmarks.

Technical Requirements

Frontend	Next.js, Material UI, WebRTC
Backend	Node.js, Express.js
AI Processing	OpenAI GPT-4-turbo, LangChain
Video Streaming	LiveKit SDK, HeyGen API
Storage	PostgreSQL, FAISS (Vector DB)

Tools & Libraries (Low-Level System Design)

AI Processing	OpenAI GPT-4-turbo
AI Memory & Context	LangChain Memory
Video Communication	LiveKit SDK
AI Avatar	HeyGen API
Resume Parsing	OpenAI Embeddings
Candidate Search	FAISS / ChromaDB

User Flow - Refer to https://mercor.com/

1. Candidate logs in.
2. Uploads resume (Extracts skills & experience).
3. AI interview begins (LiveKit creates a video room).
4. AI Avatar (HeyGen) conducts interview, adapting questions.
5. AI evaluates answers in real-time.
6. AI provides feedback & analytics.

Deployment Guidelines

- Frontend: Deploy using Vercel.
- Backend: AWS Lambda (Serverless) or DigitalOcean.
- Database: PostgreSQL with AWS RDS.

Non-Functional Requirements

- Scalability: Handle 1000+ interviews concurrently.
- Security: JWT Authentication, encrypted resume storage.
- Low Latency: Responses < 500ms using Edge Functions.

Test Cases to Follow

Upload resume	Resume parsed correctly
AI Avatar question generation	Contextually relevant questions
Real-time AI response	<500ms response time
Video latency	<200ms delay
AI scoring & feedback	Meaningful, structured feedback

Support Documentation Links

📌 LiveKit Docs: LiveKit
📌 HeyGen AI Avatar: HeyGen API
📌 OpenAI GPT-4: OpenAI Docs
📌 LangChain AI Assistant: LangChain
📌 FAISS for AI Search: FAISS Docs

Task 2: AI-Driven Job Posting, Scheduling & Recruiter Assistant

Goal of the Task

Develop an AI-powered recruitment assistant that:

- Generates job descriptions using OpenAI GPT-4.
- Posts JDs to LinkedIn & Indeed via API.
- Chats with recruiters to refine JDs before posting.
- Shortlists candidates from LinkedIn API.
- Schedules interviews via OpenAI & WhatsApp.

Architecture Overview

1. Frontend: Next.js UI for recruiters to interact with AI.
2. Backend: Node.js, Express.js for API calls.
3. AI Models: OpenAI GPT-4-turbo for JD generation.
4. Job Platform APIs: LinkedIn, Indeed for posting JDs.
5. Candidate Processing: Fetch applications, rank via FAISS.
6. Scheduling & Notifications: Google Calendar, WhatsApp API.

Key Functionalities to Achieve

1. Generate JDs using OpenAI GPT-4-turbo.
2. Post JD to LinkedIn/Indeed via API.
3. AI Chat for recruiters to refine JDs.
4. Fetch & shortlist candidates via LinkedIn API.
5. Schedule interviews via Google Calendar, Twilio WhatsApp API.

Technical Requirements

JD Generation	OpenAI GPT-4-turbo
AI Chatbot	OpenAI Assistant API
Candidate Shortlisting	FAISS, LangChain
API Integration	LinkedIn, WhatsApp, Google Calendar
Deployment	Vercel, AWS Lambda

User Flow

1. Recruiter logs in.
2. Requests JD generation via OpenAI.
3. Reviews JD with AI assistant.
4. AI posts JD to LinkedIn/Indeed.
5. AI fetches candidates & shortlists top matches.
6. Interviews scheduled via WhatsApp & Google Calendar.

Test Cases to Follow

AI JD Generation	JD generated in <100ms
Recruiter chat	AI provides meaningful JD refinements
Job posting	JD posted successfully via API
Candidate shortlisting	Candidates ranked by skill match
Interview scheduling	WhatsApp reminder sent

Support Documentation Links

📌 LinkedIn API: LinkedIn API
📌 WhatsApp API: Twilio WhatsApp
📌 Google Calendar API: Google Calendar

Evaluation Criteria & Expected Output from Candidate Submission

Task 1: AI-Powered Interview Application (LiveKit + OpenAI)

Evaluation Criteria

LiveKit Integration	20%	Correct setup of LiveKit video rooms, handling real-time streams.
OpenAI Assistant Integration	20%	AI-driven dynamic interview questions and responses.
Resume Parsing & AI Personalization	15%	OpenAI embeddings used to extract skills and personalize the interview.
LangChain Implementation	15%	Context-aware AI memory & conversation handling.
FAISS/ChromaDB Search	10%	AI-based similarity search for past interviews.
System Performance & Scalability	10%	Ensuring <500ms AI response time, handling 1000+ concurrent interviews.
Code Modularity & Best Practices	10%	Clean, structured, and well-documented code.

Expected Output from Candidate Submission

Candidates must submit the following artifacts:

1. **AI-Powered Interview System Codebase** (GitHub Repository)
    - Next.js frontend
    - Node.js/Express backend
    - LiveKit integration
    - OpenAI API implementation
    - LangChain memory management
2. **README.md**
    - Setup instructions
    - API keys configuration
    - Architecture overview
3. **Postman Collection**
    - API endpoints for LiveKit, OpenAI, and resume processing
4. **Deployment Link (Optional)**
    - Hosted application on Vercel/AWS
5. **Unit & Integration Tests**
    - Jest test cases for API functions

Task 2: AI-Driven Job Posting, Scheduling & Recruiter Assistant

Evaluation Criteria

Job Description Generation (OpenAI)	20%	AI-generated JDs are structured and contextually relevant.
AI-Powered Recruiter Chat	20%	AI assistant refines JDs based on recruiter inputs.
LinkedIn/Indeed API Integration	15%	Automated JD posting with correct OAuth handling.
Candidate Shortlisting (FAISS/ChromaDB)	15%	AI ranks candidates based on skill-matching.
Interview Scheduling (Google Calendar, WhatsApp API)	10%	AI schedules interviews and sends reminders.
System Performance & API Rate Handling	10%	API responses <500ms, efficient job postings & scheduling.
Code Modularity & Best Practices	10%	Well-structured, readable, and maintainable code.

Expected Output from Candidate Submission

Candidates must submit the following artifacts:

1. **AI-Driven Recruiter Assistant Codebase** (GitHub Repository)
    - OpenAI-based JD generation
    - LinkedIn/Indeed API integration
    - LangChain-powered AI recruiter chat
    - Candidate shortlisting via FAISS
    - Scheduling with Google Calendar & WhatsApp API
2. **README.md**
    - API setup & authentication instructions
    - System architecture explanation
3. **Postman Collection**
    - API endpoints for JD generation, posting, shortlisting, scheduling
4. **Deployment Link (Optional)**
    - Hosted application for testing
5. **Unit & Integration Tests**
    - Test cases covering API responses, job posting, and scheduling logic