AI Engineer Assessment Task

Role: AI Engineer
Duration: 3 Days (72 hours)
Submission: GitHub repository + README + Demo Video (5–7 minutes)

1. Objective

The goal of this task is to evaluate your ability to design and implement a production-grade, multi-agent AI system with real-time interaction, modular backend architecture, and a modern frontend.

You will build an AI-powered Candidate Assessment Platform that:

- Collects candidate data from multiple sources
- Builds a unified knowledge base
- Performs chat-based and voice-based AI interviews
- Uses a multi-agent system with Human-in-the-Loop (HITL) handoff to HR

This task is intentionally scoped to test system design, agentic workflows, backend engineering, real-time AI, and applied LLM usage—not just prompt engineering.

2. High-Level System Overview

The application consists of four major layers:

A. Frontend (Next.js)

- Candidate data input
- Chat-based AI assessment UI
- Real-time avatar-based voice interview UI
- HR review dashboard

B. Backend (FastAPI)

- Modular API services
- Background processing & caching
- WebSocket-based real-time communication
- Multi-agent orchestration

C. AI Layer

- LangChain / LangGraph-based multi-agent system
- LLM of your choice
- Voice + avatar-based real-time agent

D. External Providers (Your Choice)

- LLM Provider (OpenAI, Anthropic, Groq, etc.)
- Avatar Provider (D-ID, HeyGen, LiveKit Avatar, etc.)
- Real-Time Communication (LiveKit preferred)

3. Functional Requirements

3.1 Candidate Data Ingestion

The system must accept the following inputs:

- LinkedIn Profile URL
- GitHub Profile URL
- Resume Upload (PDF)

Backend Responsibilities:

- Fetch and parse public LinkedIn & GitHub data (mocking allowed)
- Extract experience, skills, projects, education, and GitHub activity
- Parse resume PDF into structured JSON

Constraints:

- Use background tasks for ingestion
- Implement caching (Redis or in-memory)

3.2 Knowledge Base Construction

- Combine all extracted data into a unified candidate knowledge base
- Store data for RAG and agent reasoning
- Optional: Vector database (FAISS, Chroma, Weaviate)

3.3 Multi-Agent System (Core Requirement)

Required Agents:

- Profile Analyzer Agent
- Technical Interviewer Agent
- Behavioral Interviewer Agent
- Evaluation Agent
- HR Handoff Agent (Human-in-the-Loop)

Agents must communicate, remain modular, and demonstrate orchestration logic.

3.4 Chat-Based AI Assessment

- Web-based chat interface
- WebSocket-based backend
- Context-aware responses from knowledge base

3.5 Real-Time Voice Interview with Avatar

- One-to-one voice-based interview
- Live audio streaming (LiveKit preferred)
- STT → LLM → TTS pipeline
- Avatar-driven interviewer

3.6 Human-in-the-Loop (HR Review)

- AI-generated final assessment report
- HR dashboard with scores, strengths, risks, and recommendations
- Optional manual HR override

4. Technical Requirements

Backend (Mandatory):

- FastAPI
- Modular architecture
- Background tasks & caching
- WebSockets
- Error handling & logging

Frontend (Mandatory):

- Next.js
- Modern UI (Tailwind / ShadCN)
- Responsive design

AI Stack:

- LangChain or LangGraph
- Any LLM provider

5. Architecture & Code Quality Expectations

- Clean, readable, and well-structured code
- Clear separation of concerns
- Reusable components
- Environment-based configuration
- Comprehensive README

6. Restrictions & Constraints

- No hard-coded secrets
- No monolithic backend
- Mocks must be clearly documented

7. Evaluation Criteria

Area	Weight
System Design & Architecture	25%
Multi-Agent Implementation	25%
Real-Time Voice + Avatar	20%
Code Quality & Modularity	15%
UI/UX	10%
Documentation	5%

8. Deliverables

- LangGraph-based State Machine
- Evaluation Metrics Visualization
- Role-Based Access Control (Candidate vs HR)
- Dockerized Setup
- Architecture Diagram
- GitHub Repository
- README.md
- Demo Video (5–7 minutes)

Good luck. Focus on clarity, architecture, and realistic AI workflows over feature overload.