### **AI-Powered Professional Assessment Agent**

**Concept**:  
I want to develop an interactive AI agent that assists users in conducting a thorough professional assessment. The agent will analyze the user’s CV, generate relevant questions about their career path, and guide them through a dynamic self-evaluation process.

**Key Features:**

1. CV Analysis & Dynamic Questioning
   * The AI scans the user’s uploaded CV/resume and identifies key areas for exploration (skills, experiences, career gaps, motivations, etc.).
   * It generates personalized questions to help the user reflect on their professional journey.
2. Voice-Based Interaction
   * Instead of text input, the AI uses a conversational avatar (via speech synthesis) to ask questions naturally, simulating a human career consultant.
   * Users respond verbally, making the experience more intuitive and engaging.
3. Data Storage & Document Generation
   * All responses are securely stored locally for privacy.
   * After completing the assessment, the system compiles the CV, questions, and answers into a structured, downloadable report.
4. Professional Assessment Focus
   * Helps users identify their skills, strengths, career motivations, and potential growth areas.
   * Supports employees, job seekers, and public servants in defining career or training goals.

Why This Approach?

* Human-Like Interaction: Voice interaction with an AI avatar makes the process feel more natural than filling out forms.
* Actionable Insights: The final document serves as a personalized career roadmap.
* Accessibility: Voice input lowers barriers for users who prefer speaking over typing.

### **Technical Aspect**

### **Service Architecture**

#### **1. CV Analysis**

Description:  
This service takes a CV file (PDF/DOCX), processes it via an LLM, and generates a structured textual summary of the candidate’s career path, skills, and experience.

API Endpoint:

python

def cv\_analysis(file\_path: str) -> str:

"""

Input: File path to the user's CV.

Output: Textual description summarizing the CV's content.

"""

#### **2. Question Generation**

Description:  
Generates dynamic, context-aware questions based on the CV and user responses.

Sub-Services:

* First Question: Introductory question derived solely from the CV.
* Follow-up Questions: Generated based on CV summary + prior user responses.

API Endpoints:

python

def generate\_first\_question(cv\_description: str) -> str:

"""

Input: CV summary text.

Output: First question (e.g., "I see you work as a Business Consultant at JPMorgan. What career are you pursuing now?").

"""

def generate\_followup\_questions(cv\_description: str, user\_response: str) -> dict:

"""

Input: CV summary + user’s last response.

Output: JSON with next question (e.g., {"question": "What skills do you want to develop in this new role?"}).

"""

#### **3. Text-to-Speech (TTS)**

Description:  
Converts generated questions into natural speech using a third-party provider (e.g., ElevenLabs).

API Endpoint:

python

def text\_to\_speech(text: str) -> bytes:

"""

Input: Question text.

Output: Audio file (e.g., MP3) for playback.

"""

#### **4. Speech-to-Text (STT)**

Description:  
Transcribes the user’s spoken responses into text via a third-party API (e.g., OpenAI Whisper, Google Speech-to-Text).

API Endpoint:

python

def speech\_to\_text(audio\_file: bytes) -> str:

"""

Input: Recorded audio from the user.

Output: Transcribed text.

"""

#### **5. Document Generation**

Description:  
Compiles the CV summary, questions, and user responses into a professionally formatted PDF report using an LLM.

API Endpoint:

python

def generate\_document(cv\_description: str, qa\_pairs: list[dict]) -> bytes:

"""

Input: CV summary + list of {"question": str, "answer": str}.

Output: PDF file for download.

"""

### Key Notes

* API Framework: All services expose REST endpoints via FastAPI.
* Modularity: Each service is decoupled for scalability (e.g., swapping TTS/STT providers).
* Privacy: User data (audio, responses) is processed locally where possible.

Example Flow:

1. User uploads CV → cv\_analysis() → CV summary.
2. AI generates Q1 → text\_to\_speech() → user hears question.
3. User speaks answer → speech\_to\_text() → answer stored.
4. Repeat until all questions answered → generate\_document() → PDF report.