Prueba Técnica de Ingeniería de IA

Motor de Respuestas para Soporte Técnico con Entrada y Salida en Audio

Duración: 3 días

Modalidad: Remoto

Nivel: Intermedio-Avanzado

Contexto de la prueba

Empresa ficticia: "SoftHelp Inc."

SoftHelp Inc. es una compañía SaaS que ofrece plataformas educativas y de productividad para usuarios en todo el mundo. Su equipo de soporte técnico desea implementar un sistema inteligente que permita a los usuarios resolver sus problemas sin la necesidad de contactar directamente a un agente humano.

Los usuarios deben poder enviar sus preguntas a través de **audio, texto o imagen**. El sistema debe entender la pregunta, buscar información en su base de conocimiento (FAQ, manuales, etc.), generar una respuesta técnica clara y devolverla tanto por **texto como en audio generado por IA (voz)**.

Tu rol en esta prueba es **construir ese backend inteligente**.

Objetivo General

Desarrollar una **API REST multimodal** que pueda:

1. Recibir preguntas por **audio**, **texto** o **imagen**.
2. Transcribir audio a texto (**Speech-to-Text** obligatorio).
3. Buscar en una base de conocimiento (**RAG**) la información relevante.
4. Procesar pantallazos o capturas (**V2V** / OCR).
5. Generar una respuesta técnica (**LLM**).
6. Convertir esa respuesta a voz (**Text-to-Speech** obligatorio).
7. Devolver la respuesta en texto y audio.

Flujo del sistema

```
None
Entrada: audio, texto o imagen
   ↓
Transcripción (si es audio)
   ↓
RAG: búsqueda en base de conocimiento
   ↓
OCR o visión (si hay imagen)
   ↓
LLM genera respuesta
   ↓
Respuesta se entrega en:
   → Texto (JSON)
   → Audio (.mp3)
```

Requisitos Obligatorios

- El sistema debe aceptar entradas en los siguientes formatos:
  - **Audio**: archivo `.mp3` o `.wav`
  - **Texto**: string JSON
  - **Imagen**: archivo `.jpg` o `.png`
- La transcripción de audio debe hacerse con herramientas como **Whisper**.
- La generación de audio debe realizarse con **TTS open source** (`gTTS`, `Coqui TTS`, `pyttsx3`, etc.).
- La recuperación de contexto debe hacerse con **RAG** (LangChain, FAISS, ChromaDB, etc.).
- La generación de texto debe realizarse con un modelo LLM (como `mistral`, `phi`, `llama2`, etc.).
- El sistema debe exponer al menos los siguientes endpoints:

```
None
POST /support          # Entrada por texto + imagen
POST /support/audio    # Entrada por audio + imagen
```

Entrada y salida esperada

Ejemplo de entrada (formato audio + imagen):

```
None
POST /support/audio
FormData:
  audio: pregunta_usuario.mp3
  image: pantalla_error.jpg
```

Ejemplo de salida:

```
None
{
  "transcription": "Mi pantalla se queda en negro al iniciar",
  "answer": "Por favor verifique si su tarjeta gráfica está bien
conectada.",
  "audio_url": "http://localhost:8000/static/respuesta.mp3",
  "source_documents": [
    "FAQ_General.txt",
    "Manual_Usuario.pdf"
  ]
}
```

Stack recomendado (open source)

Componene	Herramienta sugerida
API REST	FastAPI
STT	Whisper (obligatorio)
TTS	gTTS, Coqui TTS (obligatorio)
RAG	LangChain + FAISS / Chroma
LLM	HuggingFace (Mistral, LLaMA, Phi)
OCR / V2V	pytesseract, CLIP, Donut

Base de conocimiento para todos los candidatos

Todos los candidatos deben usar el mismo archivo base.

> **Archivo:** `BaseConocimiento_Soporte.zip`
>
> Contiene:
>
> - `Manual_Usuario.pdf`
> - `FAQ_General.txt`
> - `Errores_Comunes.txt`
> - `imagenes/` (pantallazos simulados de errores)

Este archivo será proporcionado al inicio de la prueba y debe usarse como fuente exclusiva para el sistema RAG.

Entregables

- Repositorio GitHub o archivo `.zip` con:
  - Código fuente
  - `README.md` (setup local y Docker)
  - Dockerfile funcional
- Archivo MP3 generado por cada ejemplo
- (Opcional) video o GIF demostrando el sistema

Plazo

Tienes 3 días calendario desde el momento de entrega del archivo `BaseConocimiento_Soporte.zip` y este documento.

Buena suerte.

AI Engineering Technical Challenge

Support Response Engine with Audio Input and Output

Duration: 3 Days

Mode: Remote

Level: Intermediate–Advanced

Challenge Context

Fictional Company: "SoftHelp Inc."

SoftHelp Inc. is a SaaS company that provides educational and productivity platforms to users worldwide. Their technical support team wants to implement an intelligent system that allows users to solve their issues without contacting a human agent.

Users must be able to submit their questions via **audio, text, or image**. The system should understand the question, search its knowledge base (FAQs, manuals, etc.), generate a clear technical response, and return it both as **text and AI-generated voice**.

Your role in this challenge is to **build that intelligent backend**.

General Objective

Develop a **multimodal REST API** capable of:

1. Receiving questions via **audio**, **text**, or **image**.
2. Transcribing audio to text (**mandatory Speech-to-Text**).
3. Searching a knowledge base (**RAG**) for relevant context.
4. Processing screenshots (**V2V** / OCR).
5. Generating a technical response (**LLM**).
6. Converting that response to voice (**mandatory Text-to-Speech**).
7. Returning the response in both text and audio.

System Flow

```
Input: audio, text, or image
   ↓
Transcription (if audio)
   ↓
RAG: knowledge base search
   ↓
OCR or vision (if image)
   ↓
LLM generates a response
   ↓
Response delivered as:
   → Text (JSON)
   → Audio (.mp3)
```

Mandatory Requirements

- The system must accept inputs in the following formats:
  - **Audio**: `.mp3` or `.wav` file
  - **Text**: JSON string
  - **Image**: `.jpg` or `.png` file
- Audio transcription must be implemented using tools like **Whisper**.
- Audio generation must be implemented using **open source TTS** (`gTTS`, `Coqui TTS`, `pyttsx3`, etc.).
- Context retrieval must use **RAG** (LangChain, FAISS, ChromaDB, etc.).
- Text generation must use an LLM (e.g., `mistral`, `phi`, `llama2`, etc.).
- The system must expose the following endpoints at minimum:

```
POST /support          # Input: text + image
POST /support/audio    # Input: audio + image
```

Expected Input and Output

Sample input (audio + image):

```
None
POST /support/audio
FormData:
  audio: user_question.mp3
  image: error_screenshot.jpg
```

Sample output:

```
None
{
  "transcription": "My screen goes black on startup",
  "answer": "Please verify if your graphics card is properly
connected.",
  "audio_url": "http://localhost:8000/static/response.mp3",
  "source_documents": [
    "FAQ_General.txt",
    "User_Manual.pdf"
  ]
}
```

Recommended Tech Stack (Open Source)

Componet	Recommended Tool
REST API	FastAPI
STT	Whisper (mandatory)
TTS	gTTS, Coqui TTS (mandatory)
RAG	LangChain + FAISS / Chroma
LLM	HuggingFace (Mistral, LLaMA, Phi)
OCR / V2V	pytesseract, CLIP, Donut

Common Knowledge Base for All Candidates

All candidates must use the same base file.

> **File:** `BaseConocimiento_Soporte.zip`
>
> Contains:
>
> - `User_Manual.pdf`
> - `FAQ_General.txt`
> - `Common_Errors.txt`
> - `images/` (simulated error screenshots)

This file will be provided at the start of the challenge and must be used as the exclusive source for the RAG system.

Deliverables

- GitHub repository or `.zip` archive including:
  - Source code
  - `README.md` (local and Docker setup)
  - Functional Dockerfile
- MP3 file generated for each example
- (Optional) video or GIF demo showing the system

Deadline

You have 3 calendar days from the delivery of the `BaseConocimiento_Soporte.zip` file and this document.

Good luck.