Take Home Assessment - Elimai.ai Backend Engineer Internship

QUESTION 1:(CODING)

Build a backend tool that utilizes Azure's free Speech-to-Text API to convert doctor voice dictations into structured medical data. The backend system should transcribe real-time or uploaded voice input and map the results to predefined medical terms and services based on a mock hospital database or Excel template stored in the backend. The extracted values should be returned in structured JSON format and designed in such a way that they can later be used to populate an EMR system or exported to Excel.

The backend must expose an API that:

Accept audio uploads or audio stream triggers (You can use your microphone to feed the input)

• Transcribe audio using free Azure Speech-to-Text API.

• Parse the transcription and identify key medical entities (e.g., procedure, lab test, diagnosis).

• Match transcribed phrases with internal codes from the template or database (e.g., CBC → Complete Blood Count, code: LAB023).

• Return the structured results in JSON.

Backend Functionality

Can be implemented in any language of your choice (FastAPI / Flask / Django REST). We prefer Python

• Utilize Azure's Speech-to-Text API (Free Tier) for transcription.

• Handle multiple audio inputs or streamed speech (asynchronous if applicable).

• Support a mock template or Excel file storing:

• Medical procedure names (Lab tests, Diagnosis terms, Internal hospital codes)

• Provide a REST API that:

• Accepts .mp3/.wav files or a live speech capture trigger.

• Transcribes audio to text and extracts structured terms based on the template.

Evaluation Criteria

Correct API Implementation: Should accept audio files and return accurate JSON mapped against the template.

• Speech Accuracy & Matching: Use Azure’s API effectively and match dictation to the correct internal codes.

• Code Quality & Documentation: Clean, modular, and well-commented code with a clear README.

• Error Handling & Validation: Graceful handling of invalid audio, unmatched terms, or transcription failures.

• Bonus: A basic UI demo using JavaScript that records or uploads audio and displays the structured output.

Submission Guidelines
Push the code to a GitHub repository and share the link. Include a README with instructions on How to set up & run the backend, how to test the API using Postman and Optionally how to run the UI demo if you get to build a basic UI in JavaScript.

QUESTION 2:(CODING)

Build a backend system that utilizes Azure's free OCR API / Gemini’s free OCR API to extract relevant values from multiple PDF documents simultaneously based on a predefined template (Excel sheet) stored in the backend. The extracted values should be structured in a way that they can be later used to populate an Excel sheet. The backend must expose an API that:

Accepts PDF uploads.
Extracts key values based on a predefined excel template (feel free to create your own template, see the shared example. Your task at hand is to match these elements present in the excel to searched through when a pdf is uploaded so its corresponding values can be fetched
Returns the extracted values in JSON format.

Backend Functionality

Must be implemented in Python (FastAPI/Django Flask /REST API).
Utilize Azure’s free OCR API for text extraction.
Handle multiple PDFs at the same time (asynchronous processing).
Extract values based on a stored template that contains field names and expected text positions.

Expose a REST /Fast/Django API that:
Accepts PDF files as input.
Extracts relevant values based on the template.
Returns structured JSON output.
API Example in Postman
POST /upload (Uploads PDFs & extracts values).

Minimal Frontend (Optional, Bonus)
Use HTML + JavaScript /React/Choice of yours to create a basic UI that:
Allows PDF uploads via drag & drop.
Calls the backend API to extract values.
Displays extracted values in a simple UI.

Evaluation Criteria
Correct API Implementation: Should work as expected when tested via Postman.
Code Quality & Documentation: Clean, well-documented code with README instructions
Error Handling & Validation: Handle missing fields, incorrect formats, API failures
Bonus: Simple UI to show a working live demo

Submission Guidelines

Push the code to a GitHub repository and share the link. Include a README with instructions on How to set up & run the backend, how to test the API using Postman and Optionally how to run the UI demo if you get to build a basic UI in JavaScript.

Expecting your approach towards the problem. Don't feel nervous. Good luck.

Excel sheet template image: