Full Stack Developer Assessment: Designing an AI-Powered Document Insight Tool

1. Introduction: The Challenge Ahead

This test is designed to understand your flow of thought and toolchain familiarity. Feel free to use any tools that you usually use for development. The objective is to understand your proficiency in coding and your interest in AI.

2. Project Overview: Building an AI-Powered Document Insight Tool

The core task is to develop an AI-powered document insight tool. This application will allow users to upload PDF documents (primarily resumes) and receive a concise summary or key insights. A crucial feature will be maintaining a historical record of uploaded documents and their analyses, providing a seamless user experience. The project assesses the ability to integrate backend processing with a responsive frontend, considering architecture for data management and external service interaction.

3. Core Task: Technical Requirements

The project is divided into distinct technical components.

3.1. Backend Service Development

The application's foundation is a robust backend service for handling document uploads, orchestrating content processing, and managing data.

Server Framework Choice

Establish a web server using a Python framework. Options include Flask, Django, or FastAPI. FastAPI is suggested for those less familiar.

API Endpoints

The backend service must expose two distinct API endpoints:

- `/upload-resume` **Endpoint**: For receiving PDF file uploads. The method should align with standard web practices for file submission.
- `/insights` **Endpoint**: For retrieving processed document insights. This requires a mechanism (e.g., query parameters) to identify which insights are requested,

linking to the history feature.

Content Processing and AI Integration

The backend must process uploaded PDFs and leverage external AI for summarization (Our recommendation is to use Sarvam M since it's available for free and indegenous).

Upon PDF upload, the server extracts content and uses Sarvam AI to generate a document summary. Candidates must create a free account on the Sarvam AI platform (contact ahan@onesol.in for assistance).

Implement a robust fallback mechanism: if Sarvam AI is unavailable, the server must gracefully degrade. Instead of an AI summary, it should return the top five most frequently used words within the PDF.

3.2. Frontend Web Application

A user-facing web application complements the backend.

User Interface for Document Upload

A straightforward web interface is required for users to select and upload PDF files to the backend.

Display of Processed Insights

After backend processing, the web application must dynamically display the generated summary or fallback analysis clearly. This assesses handling and rendering diverse data from the backend, adapting presentation based on the type of insight provided.

History Feature

Implement a "History" tab or section listing all previously uploaded PDF files, allowing users to revisit their insights. It would be preferable for this historical data to persist across user sessions.

4. Submission Guidelines

Candidates should package their project as a Git repository containing all source code. A comprehensive `README.md` file is essential, providing clear setup/running instructions, detailing significant architectural/design diagrams, and offering reflections on challenges or alternative approaches.