Junior Python/AI Developer Tasks

Instructions:

Please pick one of the tasks listed below and submit your solution within 10 days from the day of receiving this email. Kindly email the submissions to jibin.g-john@amphenol-fci.com

Your final submission must include:

- A short **YouTube video demo**
- One or two-slide presentation (objective & functionality)
- **GitHub repository** with:
    - Properly structured source code
    - A detailed **README.md** (explaining the approach, setup, and usage)
- **Logs or outputs** (especially LLM interactions if applicable)
- A basic **evaluation method** for your solution

1: PDF Content Labeling + Query UI

Objective:

- Extract and label all content from a PDF using any **AI model**
- Create a **UI** (using Gradio or Streamlit) to allow users to query metadata or content from the PDF

Evaluation Criteria:

- Accuracy of extraction and labeling
- UI interactivity and clarity
- LLM/API response relevance

2: Backend API with Gemini Flash 2.0 Integration

Objective:

- Build a backend service using **FastAPI** or **Flask**

- Integrate **Google Gemini Flash 2.0** for intelligent responses or actions

Evaluation Criteria:

- API structure and endpoint design
- Gemini integration quality
- Response time and reliability

3: AI Utility Tool using Gemini LLM

Objective: Build **any one** of the following using Gemini:

- A **text/article summarizer**
- A **stock or crypto analyzer** (e.g. correlate price movement with news from last 30 days)
- A **multi-step math solver** (e.g., algebra, calculus, or word problems)

Evaluation Criteria:

- LLM prompt design and chain of reasoning
- UI or output clarity
- Real-world utility

4: RAG-Based Intelligence App

Objective:

- Extract and label content from PDF or HTML or Excel or PPT or any document using an AI model
- Build a **RAG (Retrieval-Augmented Generation)** system with **Gemini**
- Allow metadata querying through a **Gradio** or **Streamlit UI**

Evaluation Criteria:

- Quality of retrieval and RAG logic
- Accuracy of Gemini responses based on retrieved data
- UI responsiveness and usability

Task 5: Agentic AI API or Server (Multi-step Reasoning)

Objective: Build an **Agentic AI backend** that:

- Perform tasks an LLM **can't do directly** by invoking tools or APIs in the background
- Handles multi-step interactions with memory of past steps

Examples:

- Calculate sum of exponential of first 6 Fibonacci numbers
- Get top OTT series and send to Telegram/email
- Track stock price and notify if it crosses a value
- Pick any other better application

Agent Workflow Example:

Query → LLM Response → Tool Call → Tool Result → Next Query → LLM Response → Tool Call → Tool Result → Final Result

Evaluation Criteria:

- Memory handling across LLM calls
- Task orchestration and modularity
- Simplicity and reliability of agent loop

In case of any queries, please contact in jibin.g-john@amphenol-fci.com