**Requirement**: We want to build a usecase solution where the objective to achieve is mentioned below:

1. access the url https://www.dgft.gov.in/CP/?opt=regulatory-updates
2. Identify the latest Notifications, Public Notices and Circulars from the following porta. These are clickable links in the website
3. In all these links, there shall be documents inside hose links in pdf format or other format. Thise need to be opened, read and summarised and then need to send an email to the admin by attaching the file with the summarized content.

**Step-by-Step Architecture**

1. **Web Scraping Module**:
   * **Objective**: Access the URL and identify the latest Notifications, Public Notices, and Circulars.
   * **Tools**: Use libraries like BeautifulSoup and Requests in Python.
   * **Process**:
     + Send an HTTP GET request to the URL.
     + Parse the HTML content to identify clickable links for Notifications, Public Notices, and Circulars.
2. **Document Retrieval Module**:
   * **Objective**: Open and read documents in PDF or other formats.
   * **Tools**: Use libraries like PyPDF2 for PDFs and other relevant libraries for different formats.
   * **Process**:
     + For each identified link, send an HTTP GET request to retrieve the document.
     + Save the document locally for processing.
3. **Document Summarization Module**:
   * **Objective**: Summarize the content of the documents.
   * **Tools**: Use Natural Language Processing (NLP) libraries like spaCy or transformers from Hugging Face.
   * **Process**:
     + Extract text from the documents.
     + Use NLP techniques to generate a concise summary of the content.
4. **Email Notification Module**:
   * **Objective**: Send an email to the admin with the summarized content and attach the original document.
   * **Tools**: Use libraries like smtplib for sending emails.
   * **Process**:
     + Compose an email with the summarized content.
     + Attach the original document.
     + Send the email to the admin.

**Workflow Diagram**

1. **Web Scraping Module**:
   * Access URL → Identify Links → Retrieve Documents
2. **Document Retrieval Module**:
   * Open Document → Extract Text
3. **Document Summarization Module**:
   * Summarize Text
4. **Email Notification Module**:
   * Compose Email → Attach Document → Send Email

**Implementation Tips**

* **Error Handling**: Ensure robust error handling at each step to manage network issues, document retrieval failures, and email sending errors.
* **Logging**: Implement logging to track the progress and any issues encountered during the process.
* **Scheduling**: Use a task scheduler like cron (Linux) or Task Scheduler (Windows) to run the script at regular intervals.

Would you like more details on any specific part of this architecture?

====================== Claude =============================

dgft\_regulatory\_monitor/

│

├── .env # Environment variables (API keys, URLs)

├── main.py # Main entry point

├── requirements.txt # Project dependencies

│

├── agents/

│ ├── \_\_init\_\_.py

│ ├── web\_scraper\_agent.py # Web scraping and navigation

│ ├── pdf\_extractor\_agent.py # PDF extraction and downloading

│ ├── ocr\_agent.py # OCR processing for image-based PDFs

│ ├── analysis\_agent.py # LLM-based analysis and insights

│ └── coordinator\_agent.py # Orchestrates the workflow between agents

│

├── utils/

│ ├── \_\_init\_\_.py

│ ├── date\_utils.py # Date comparison and processing

│ ├── pdf\_utils.py # PDF processing utilities

│ └── logging\_utils.py # Logging functionality

│

└── ui/

├── \_\_init\_\_.py

└── app.py # Streamlit frontend