Technical Document: Automated Email PDF Processing and JSON Data Extraction

# Table of Contents

1. Project Overview
2. Project Goals
3. System Architecture
4. Components and Tools Used
5. Detailed Workflow
6. Technical Details
   * Email Fetching
   * PDF Processing
   * JSON Data Generation
7. Error Handling and Logging
8. Future Enhancements
9. Conclusion

# 1. Project Overview

This project aims to automate the process of fetching PDF attachments from emails, extracting text from these PDFs, and generating structured JSON data from the extracted text. This system is designed to handle large volumes of emails and PDFs efficiently, providing a robust solution for data extraction and processing.

# 2. Project Goals

* **Automate Email Fetching**: Automatically fetch emails with specific subjects and download PDF attachments.
* **Extract Text from PDFs**: Use OCR to convert PDF content into text format.
* **Generate JSON Data**: Extract relevant data from the text and structure it into JSON format based on predefined schemas.
* **Real-time Processing**: Ensure the system operates in real-time, processing emails and PDFs as they are received.

# 3. System Architecture

The system comprises several interconnected components, each responsible for a specific task in the workflow. The architecture can be visualized as follows:

1. **Email Fetching**: Retrieve emails and download attachments.
2. **PDF Processing**: Convert PDF attachments to text using OCR.
3. **JSON Generation**: Extract data from text and structure it into JSON.

# 4. Components and Tools Used

* + **Programming Language**: Python
  + **Libraries**:
    - **`imaplib`** and **`email`**: For fetching and processing emails.
    - **`pdf2image`** and **`pytesseract`**: For converting PDFs to images and extracting text using OCR.
    - **`json`**: For handling JSON data.
    - **`dotenv`**: For environment variable management.
    - **`langchain`**: For creating extraction chains.
    - **`schedule`** and **`watchdog`**: For scheduling tasks and monitoring directories.
  + **External Services**:
    - **Ollama Functions**: For running language models.

# 5. Detailed Workflow

## Email Fetching

1. **Connect to Email Server**: Use `imaplib` to connect to the Gmail server.
2. **Login and Select Mailbox**: Authenticate using credentials and select the inbox.
3. **Search and Fetch Emails**: Retrieve emails with the subject "Invoice" and download PDF attachments.

## PDF Processing

1. **Convert PDFs to Images**: Use `pdf2image` to convert PDF pages to images.
2. **Extract Text Using OCR**: Apply `pytesseract` to extract text from images.
3. **Save Extracted Text**: Store the extracted text for further processing.

## JSON Data Generation

1. **Define Data Schemas**: Use predefined schemas for user data, product data, supplier data, and product items.
2. **Extract Data**: Apply `langchain` to extract data from text based on the schemas.
3. **Combine and Save JSON**: Merge the extracted data into a single JSON object and save it.

# 6. Technical Details

## Email Fetching

* + **File**: `email\_fetch.py`
  + **Description**: Connects to the Gmail server, searches for emails with the subject "Invoice," and downloads PDF attachments to a specified directory.

## PDF Processing

* + **File**: `pdf\_processor.py`
  + **Description**: Converts PDF files to images, extracts text using OCR, and processes the text to generate JSON data.
  + **Functions**:
    - **`read\_pdf(file\_path) `**: Converts PDF to images and extracts text.
    - **`process\_pdfs\_in\_folder (input\_folder, output\_folder) `**: Processes all PDFs in a given folder and generates JSON files.

## JSON Data Generation

* + **File**: `PO\_json.py`
  + **Description**: Defines schemas for extracting data from text and combines the extracted data into a single JSON object.
  + **Functions**:
    - **`combine\_json\_objects(\*args) `**: Merges multiple JSON objects.
    - **`gen\_json(text\_data) `**: Generates JSON data from extracted text.

## Language Model (LLM)

* + **Model**: llama3.1-latest
  + **Description**: The llama3.1-latest model is used for extracting structured data from unstructured text. It leverages advanced natural language processing (NLP) capabilities to understand and extract relevant information based on predefined schemas.
  + **Integration**: The model is integrated using langchain and Ollama Functions, which allow for the creation of extraction chains tailored to the specific data schemas required for the project.

# 7. Error Handling and Logging

* + **Email Fetching**: Checks for connectivity and authentication errors, logs appropriate messages, and exits gracefully.
  + **PDF Processing**: Handles file reading errors, connection issues with the model server, and key errors during JSON generation.
  + **Logging**: Implement logging at various stages to capture errors and processing status.

# 8. Future Enhancements

* + **Enhanced Error Handling**: Improve error handling mechanisms and add more detailed logging.
  + **Scalability**: Optimize the system for handling larger volumes of emails and PDFs.
  + **User Interface**: Develop a user-friendly interface for monitoring and managing the processing pipeline.
  + **Integration**: Integrate with other data sources and systems for more comprehensive data extraction and processing.

# 9. Conclusion

This project provides an automated solution for email PDF processing and JSON data extraction. By leveraging various Python libraries and external services, the system ensures efficient and accurate data processing. Future enhancements will focus on improving scalability, error handling, and user experience.