Gen AI-Based Email Classification and OCR

Problem statement:

Commercial Bank Lending service teams receive a high volume of servicing requests via email. These emails often contain diverse requests and attachments, which are ingested into the loan servicing platform to create service requests (SRs). These SRs then go through workflow processing.

The challenge is to automate email classification and data extraction using Generative AI (LLMs), improving efficiency, accuracy, and turnaround time while minimizing manual intervention.

Inspiration:

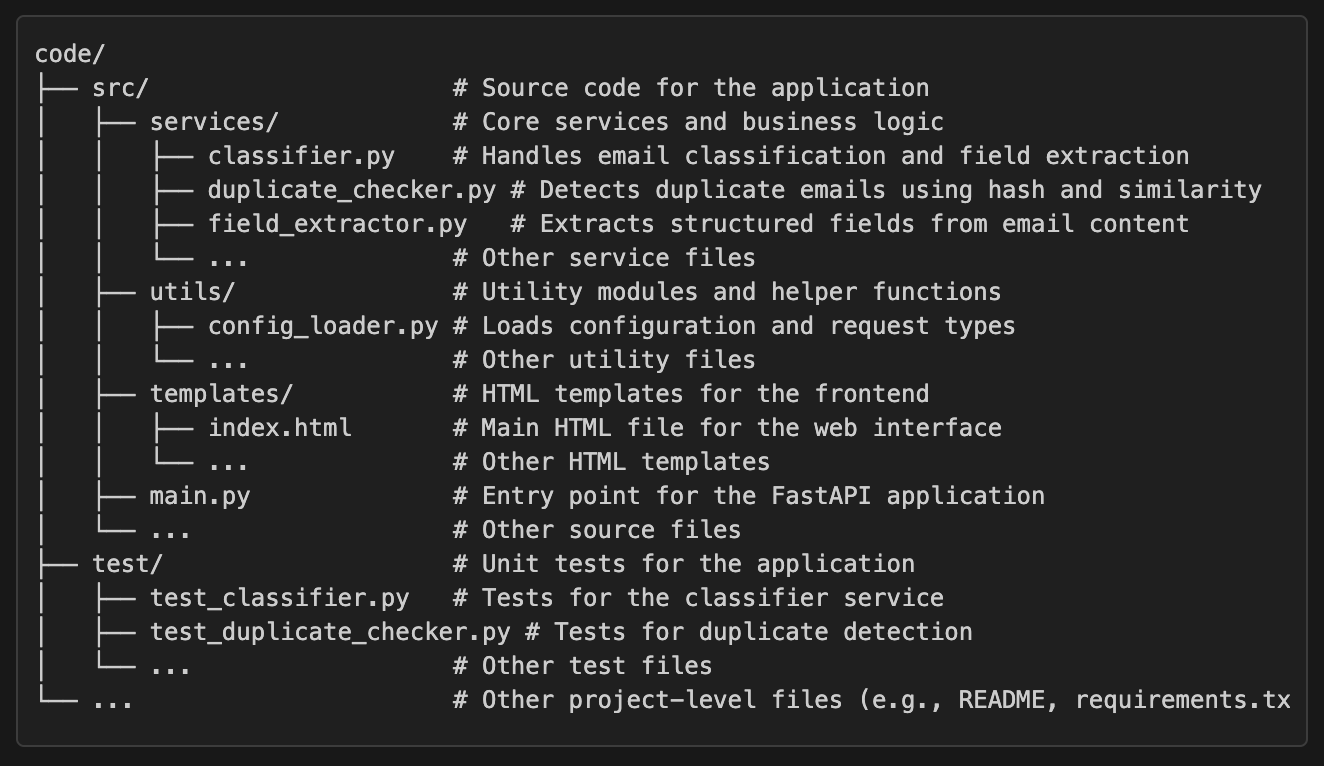
Manual email triage is time-consuming, inefficient, and prone to errors. Automating this process with AI can drastically improve accuracy and reduce turnaround time.

List of Data Source/Files Accepted:

Emails - .eml files with attachments

Attachments - PDFs, DOCX

Folder Structure:



Solution Flow:

#### **1. Email & Attachment Input**

* The system receives an email with or without attachments.
* Attachments can be PDFs, Word documents, or other supported formats.

#### **2. Ingestion & Preprocessing**

* The email and attachments are ingested into the system.
* Preprocessing is performed, including:
  + Removing unnecessary headers and footers.
  + Standardizing text formatting.
  + Handling encoding issues.

#### **3. Email/Attachment Type Classification**

* The system determines whether the input is:
  + **Email (.eml)** → Proceed to email parsing.
  + **Attachment (.pdf, .docx, etc.)** → Proceed to attachment parsing and OCR.

#### **4A. Email Parsing**

* Extract structured email content:
  + Sender, receiver, subject, body, timestamps, etc.
* Identify quoted text (replies/forwards) and extract relevant portions.

#### **4B. Attachment Parsing & OCR**

* If an attachment is present, extract text from:
  + **PDF, DOCX:** Use standard text extraction methods.
  + **Scanned PDFs/Images:** Use OCR (PaddleOCR) to extract text.

#### **5. Text Extraction**

* Consolidate extracted text from both email body and attachments.
* Prepare the extracted data for further processing.

#### **6. LLM Prompt Generation**

* Format the extracted text into a structured prompt for the LLM.
* Include:  
  + Request type definitions.
  + Custom extraction rules.
  + Contextual information from past classified emails.

#### **7. LLM Processing (Hugging Face Requests)**

* Use **HTTP Requests** to interact with the LLM (Mistral-7B)
* The model analyzes the email content and extracted text to:  
  + Classify the request type.
  + Identify relevant sub-request types.
  + Determine key extracted fields.

#### **8. Request Type/Data Extraction/Duplicate Detection**

* The system performs three key tasks:
  1. **Request Type & Sub-Request Type Identification:**
     + LLM classifies the request based on historical data and definitions.
  2. **Data Extraction:**
     + Extract required fields like deal name, amount, expiration date, etc.
  3. **Duplicate Detection:**
     + Check if the email is a duplicate of an existing request using hashing-based methods.

#### **9. Classification & Confidence Scoring**

* Assign a confidence score to the classification and extracted fields.
* If confidence is low, flag for manual review.

#### **10. Output Formatting - JSON**

Structure the extracted data in JSON format:  
  
{

"request\_type": "Money Movement Inbound",

"sub\_request\_type": "Principal",

"extracted\_fields": {

"deal\_name": "XYZ Deal",

"amount": "$500,000",

"expiration\_date": "2025-04-01"

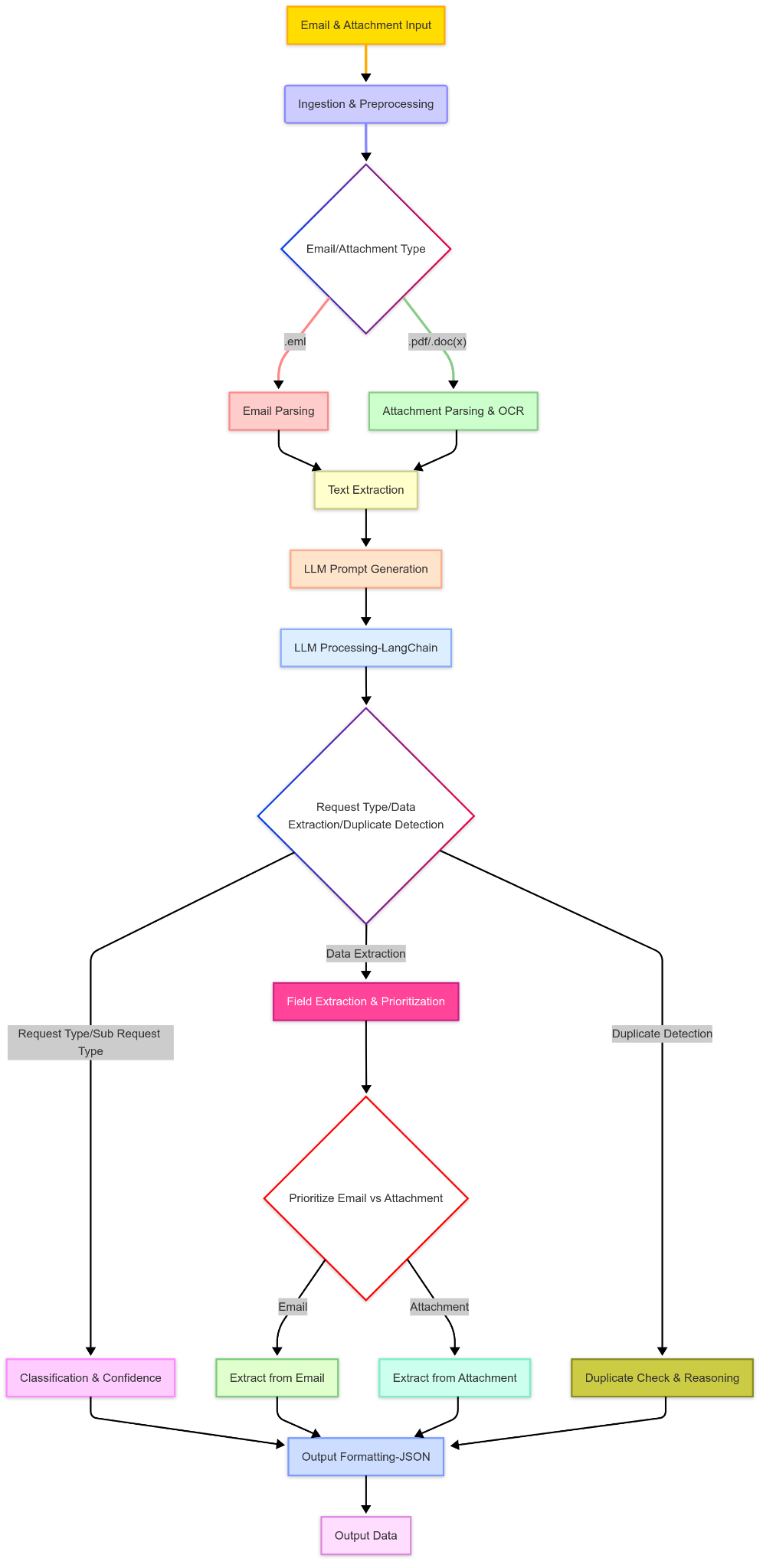
},

"confidence\_score": 0.92,

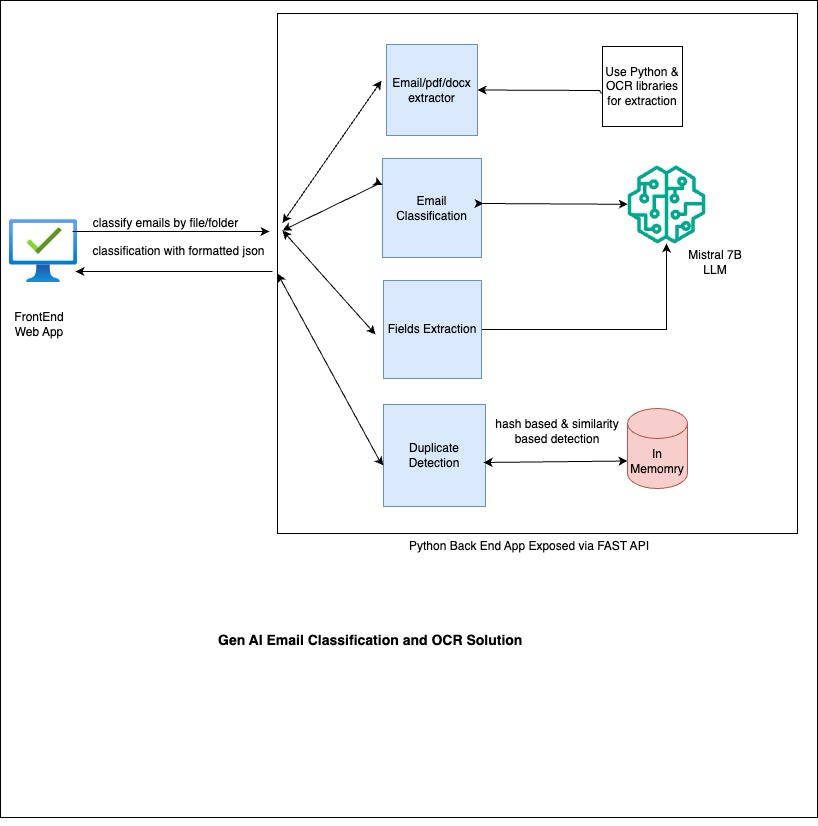
"duplicate\_flag": false

}

Flow Diagram Can be found in next page



Architecture Diagram:



**Tech Stack :**

* 🔹 Frontend: React, BootStrap, HTML, CSS
* 🔹 Backend: Python Fast API
* 🔹 Database: In memory, Chroma DB
* 🔹 AI & NLP : Mistral-7B , scikit-learn
* 🔹 Document Processing : eml-parser, paddleocr ,pdfplumber,python-docx
* 🔹 Frontend: React, BootStrap, HTML, CSS
* 🔹 Backend: Python Fast API
* 🔹 Database: In memory, Chroma DB
* 🔹 AI & NLP : Mistral-7B , scikit-learn
* 🔹 Document Processing : eml-parser, paddleocr ,pdfplumber,python-docx