Technology Hackathon-Generative AI

**Problem Statement :**

Commercial Bank Lending Service teams handle a high volume of servicing requests via email, often with attachments. These emails require manual triage by gatekeepers who read, classify request types, extract key details, and assign tasks to the right teams. This process is time-consuming, error-prone, and inefficient, especially at scale.

The challenge is to automate email classification and data extraction using Generative AI (LLMs) to improve accuracy, efficiency, and turnaround time while minimizing manual effort. The solution should classify emails, extract relevant information for service requests, and enable skill-based auto-routing to the appropriate processing teams.

**Tools / TechStack Used:**

1. Python
2. Frontend - Streamlit
3. LLM- sambanova/Llama-3.1-Swallow-8B-Instruct-v0.3, groq/llama3-8b-8192
4. CrewAI (Agentic AI)
5. GMail API -python client
6. Text Extraction   
    1. Pdfplumber- Document  
    2. Pptx - Powerpoint  
    3. Pandas - Excel  
    4. PyTesseract- Image
7. Vectore store – Fiass cpu

8 Embedding model - SentenceTransformer("all-MiniLM-L6-v2")

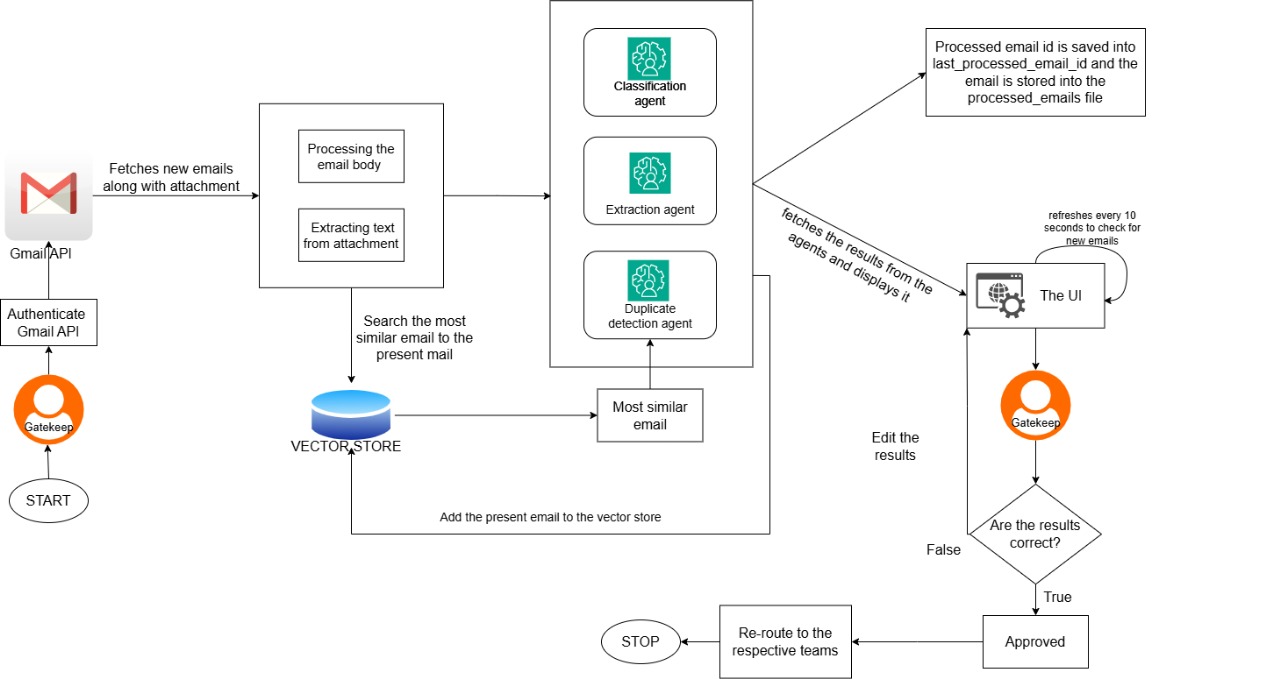
**Solution Proposed :**

The proposed solution is an AI-powered email processing system that automates the extraction, classification, and analysis of email content, including attachments. The system fetches emails from a Gmail inbox using the Gmail API, processes the email body along with its attachments (PDFs, images, PowerPoint slides, and Excel sheets), and then performs intelligent classification using LLM’s and GenAI Agents.

To enhance efficiency, the system integrates Optical Character Recognition (OCR) for extracting text from images and scanned PDFs, while libraries like PyMuPDF and Pandas process textual PDFs and Excel sheets. The extracted content is then combined with the email text and passed through a model for categorization and prioritization.Using Retrieval Augmented Generation duplicate documents are figured out.

The architecture ensures scalability by supporting large volumes of emails and attachments while maintaining high processing speed. The uniqueness of the solution lies in its ability to understand email intent holistically by merging structured (Excel, PPT) and unstructured (image, text) data for a unified analysis. The efficiency is maximized by parallel processing and optimized data pipelines, ensuring real-time insights for business decision-making.

**Architecture Diagram :**



The architecture follows a modular and scalable design:

1. Email Fetching & Authentication

Start: The process begins with the Gatekeeper (user authentication system).  
 Authenticate Gmail API: The system connects to Gmail using OAuth authentication.  
 Fetch Emails: The system retrieves new emails and attachments from Gmail.

2️. Email & Attachment Processing

Extract Email Body: Parses the email body content.  
 Extract Text from Attachments: Extracts text from PDFs, Excel, PPTs, and images.

3️. Vector Store for Similar Email Search

Search for Similar Emails: Looks up past emails in the Vector Store to find the most relevant match.  
Store Current Email: Adds the new email to the Vector Store for future reference.

4️. AI Agents Processing

Classification Agent: Categorizes the email based on its intent.  
Extraction Agent: Identifies and extracts key information (e.g., request type, customer details).  
Duplicate Detection Agent: Checks if the email is a duplicate of a previously processed one.

5️. UI & Human Verification

Display Results: The UI fetches classification results from the AI agents.  
User Verification: The Gatekeeper verifies and edits results if needed.  
Automatic Refresh: The UI refreshes every 10 seconds to check for new emails.

6️. Email Routing & Final Processing

Approved Emails: If results are correct, the email is approved.  
Incorrect Emails: If incorrect, the results are edited.  
Routing to Teams: Approved emails are forwarded to the respective teams.  
Stop: Process ends once the email is successfully routed**.**

**Folder Structure :**

hackathon

├── attachments

├── .env

├── config.py

├── credentials.json

├── crew.py

├── extractor.py

├── gmail\_service.py

├── last\_processed\_id.txt

├── main.py

├── models.py

├── processed\_emails.pickle

├── pyvenv.cfg

├── quickstart.py

├── storage.py

├── token.json

├── ui\_styles.py

**Procedure and Workflow :**

The process consists of 5 key stages:

1. Gmail Authentication & Email Fetching
2. Email Processing (Classification, Duplicate Check, Field Extraction)
3. Storing Processed Emails
4. Displaying Emails in UI
5. Auto-Refresh & Continuous Processing

**Gmail Authentication & Email Fetching :**

*Objective:* Connect to the Gmail API, authenticate using OAuth2, and fetch new loan servicing emails.

### **Step 1: Authenticate Gmail API**

* Uses quickstart.py for initial authentication & token generation.
* Reads credentials.json to authorize Gmail API.
* Stores access tokens in token.json for re-authentication.

### **Step 2: Fetch Unprocessed Emails**

* Connects to Gmail API using get\_gmail\_service().
* Fetches emails received **after the last processed email**.
* Extracts subject, sender, date, and body content.
* Uses fetch\_emails\_after\_id() to get email IDs.

## 

## **Email Processing (Classification, Duplicate Check, Field Extraction)**

*Objective:* Process the fetched emails using AI models.

### **Step 3: Extract & Process Email Data**

* Fetches email details using get\_email\_details().
* Extracts subject, sender, date, body, and snippet.
* Uses regex-based function to extract the sender email.

### **Step 4: Classify Email (CrewAI)**

* Calls CrewAI (process\_email\_with\_crew) to classify the email.There are three different types of agents  
   1.Classification Agent  
   2.Extraction Agent  
   3.Duplicate Email Detection Agent
* Identifies Request Type, Sub-Request Type, and Confidence Score.

### **Step 5: Detect Duplicate Emails**

* Marks an email as duplicate if the body text is highly similar .
* Used RAG approach to find the similarity between the documents .

### **Step 6: Extract Loan Servicing Data**

* Extracts Borrower Name, Loan Amount, Payment Date using NLP.

**Storing Processed Emails**

*Objective:* Save processed emails for future reference.

### **Step 7: Store Processed Email IDs**

* Saves last processed email ID to prevent duplicate processing.

## **Displaying Emails in UI**

*Objective:* Show emails with classification results in **Streamlit UI**.

### **Step 8: Display Processed Emails**

* Shows Subject, Request Type, Sub-Type, Confidence Score.
* Includes Accept & Edit buttons for manual verification.

## **Auto-Refresh & Continuous Processing**

*Objective:* Enable automatic fetching & processing of new emails.

### **Step 9: Auto-Refresh New Emails**

* Runs every 10-300 seconds based on UI slider.
* Calls fetch\_and\_process\_new\_emails() at intervals.

**Challenges Faced:**

### **Handling Different Attachment Formats**

To process various attachment types, format-specific parsers such as PyMuPDF, Pandas, and python-pptx were used. These tools facilitate the extraction of structured content from PDFs, spreadsheets, and PowerPoint presentations, ensuring that attachments are processed efficiently without data loss.

### **Extracting Text from Images and Scanned PDFs**

For non-editable documents, OCR technologies like Tesseract were integrated. These tools convert text from scanned images and PDFs into machine-readable formats, enhancing the system’s ability to extract relevant information accurately.

### **Efficient Email Retrieval**

To optimize the retrieval of emails, Gmail API calls were refined, and caching mechanisms were implemented. By reducing redundant API requests and temporarily storing frequently accessed data, the system improves response times and minimizes unnecessary API usage.

### **Categorizing Diverse Email Content**

A fine-tuned LLM model for multi- request classification was deployed to analyze email text. Agentic AI is incorporated to detect multiple relevant categories simultaneously with primary intent, ensuring precise classification of diverse email content**.**

### **Handling Large-Scale Email Processing**

To efficiently manage high email volumes, parallel processing techniques and optimized database queries were implemented. This approach allows for concurrent email processing, reducing execution time and ensuring smooth scalability for large datasets.

## 

## **Uniqueness & Scalability**

### **Uniqueness**

* Processes both structured (Excel, PowerPoint) and unstructured (PDFs, images) data.
* Merges email body and attachment data in a realistic fashion for holistic analysis.
* Uses AI-driven classification for intelligent email routing.

### **Scalability**

* Supports parallel processing for handling large email volumes.

## **Efficiency & Performance**

* **Optimized Execution**: Implements asynchronous API calls for fetching emails.
* **Fast Processing**: Uses efficient text extraction methods for different file types.
* **Accurate Classification**: Fine-tuned LLM and Agentic models ensure precise email categorization.

## **Conclusion**

The implemented solution efficiently processes and categorizes emails by leveraging a combination of format-specific parsers, OCR technologies, optimized API calls, fine-tuned AI models and AI agents, and scalable processing techniques. The system ensures seamless handling of diverse email formats, accurate text extraction, and intelligent classification, allowing for streamlined email processing and automation. By incorporating caching mechanisms, parallel processing, and database optimizations, the solution enhances performance and scalability, making it well-suited for large-scale enterprise applications. Overall, this architecture provides a robust and efficient framework for intelligent email classification and processing.