

# GEN AI BASED EMAIL CLASSIFICATION AND OCR



# INTRODUCTION

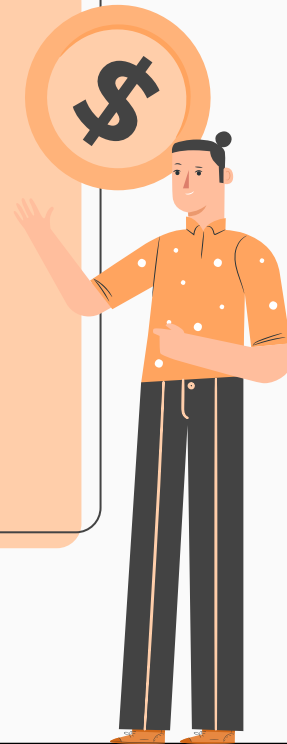


## Overview:

- \* This AI-powered tool automates email classification and data extraction using Gen AI LLMs.
- \* Enhances efficiency, accuracy, and turnaround time for processing emails.
- \* Generates classified service requests automatically.

## Key Features:

- \* Users can upload emails, PDFs, and images via a user interface.
- \* The system extracts and classifies content.
- \* Helps reduce manual effort and human errors in classification.



# INSPIRATION



## Problem Statement:

- Current manual process requires a team of gatekeepers.
- Time-consuming and prone to human errors.
- Inconsistent data extraction and classification.

## Objective:

- Automate the entire email classification and data extraction process
- Reduce human intervention while improving accuracy and speed.

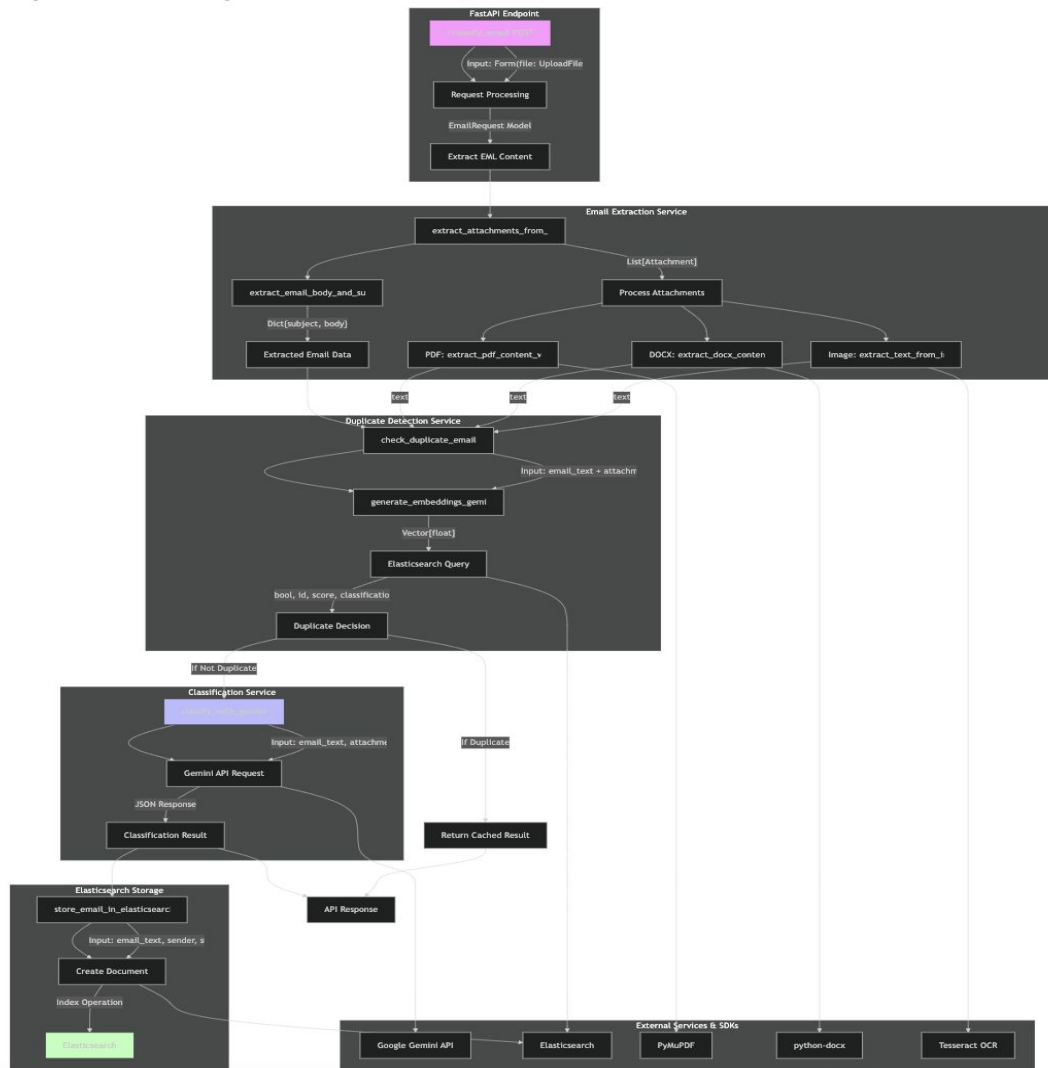


# WHAT IT DOES

- **Content Extraction** : Extracts text from emails and their attachments.
- **OCR for PDFs and Images** : Uses pytesseract and PyMuPDF for image and PDF text recognition.
- **Context-Based Data Extraction** : Uses LLMs to extract relevant data based on the email context.
- **Duplicate Detection** : Generates vector embeddings stored in Elasticsearch to identify duplicate emails.
- **Email Classification** : Categorizes emails into predefined request types using Gen AI LLMs.
- **Handling Multi-Request Emails** : Identifies primary intent when multiple requests exist in a single email.
- **Priority-Based Extraction** : Users can set priority rules for extracting important information first.
- **User Interface** : Users can upload emails and view classification responses.
- **Handling Disputes for Duplicates** : If an email is wrongly marked as a duplicate, the user can dispute it, and the system will reclassify it.



# ARCHITECTURE DIAGRAM



# HOW WE BUILT IT

## Backend:

- **FastAPI** → API framework.
- **Uvicorn** → Runs the FastAPI application
- **Elasticsearch** → Stores email data and performs similarity searches.
- **Google Gemini (GenAI)** → Text classification and embeddings.
- **Spacy** → NLP processing.
- **pytesseract** → OCR for image-based text extraction.
- **PyMuPDF (fitz)** → PDF text extraction.
- **docx library** → Extracts text from DOCX files.
- **BeautifulSoup** → Parses HTML content from emails.

## Frontend:

- **ReactJS** → UI for uploading files and viewing responses.

## Tools Used:

- **Postman** → API testing.
- **Jupyter & VSCode** → Development environment.
- **Canva** → Demo video creation.

## CHALLENGES WE FACED



- **OCR Implementation for PDFs:**

- Initially used pdf2image, but it did not convert PDFs to images properly.
- Switched to PyMuPDF and pytesseract, improving accuracy.

- **LLM Prompt Optimization:**

- Faced difficulty in getting reliable classification results.
- Optimized LLM prompts and fine-tuned responses for better accuracy.

# TECH STACK



COMPONENT	Technology Used
FRONTEND	HTML, CSS, React, TypeScript
BACKEND	Python, FastAPI, Google GenAI
DATABASE	Elasticsearch
GEN AI MODELS	LLM: "gemini-2.0-flash", Embeddings: "text-embedding-004"



# API REQUEST FORMAT

```
{  
  "file_path":"/Users/sparsh/Downloads/structured_sample_email_pdfs/email_4.pdf",  
  "sender":"sparsh",  
  "subject":"Test email",  
  "user_disputes_duplicate":true,  
  "priority_rules" : {  
    "content_weightage": 0.7,  
    "attachment_weightage": 0.3,  
    "keywords_priority": {}  
  }  
}
```

# API RESPONSE

The screenshot displays the Postman interface with a workspace named "API Network". A POST request is configured to `http://0.0.0.0:8000/classify_email`. The request body is set to "form-data" and includes two parameters: "file" (a file named "email4.eml") and "request\_data" (a JSON object). The response status is "200 OK" with a response time of 4.32 s and a size of 654 B. The response body is shown in JSON format.

**Request Details:**

- Method: POST
- URL: `http://0.0.0.0:8000/classify_email`
- Body Type: form-data
- Parameters:
  - file (File): email4.eml
  - request\_data (Text): `{ "request_type": "Money Movement (Funds Transfer)", "sub_request_type": "Money Movement: Inbound", "confidence_score": "0.8" }`

**Response Details:**

- Status: 200 OK
- Time: 4.32 s
- Size: 654 B
- Body Type: JSON

**Response Body (JSON):**

```
1 {
2   "classification": [
3     {
4       "request_type": "Money Movement (Funds Transfer)",
5       "sub_request_type": "Money Movement: Inbound",
6       "confidence_score": "0.8"
7     },
8     {
9       "request_type": "Money Movement (Funds Transfer)",
10      "sub_request_type": "Money Movement: Inbound",
11      "confidence_score": "0.8"
12    },
13    {
14      "request_type": "Money Movement (Funds Transfer)",
15      "sub_request_type": "Money Movement: Inbound",
16      "confidence_score": "0.8"
17    }
18  ],
19  "is_duplicate": false,
20  "extracted_context": {
21    "deal_name": "loan",
22    "amount": "500000",
23    "expiration_date": "60 days",
24    "subject": "Subject",
25    "currency": "EUR",
26    "principal_payment": "50000"
27  }
28 }
```

The background of the slide is a solid reddish-brown color. It is decorated with numerous coins of various denominations and types, including a large Bitcoin logo on the left side. The coins are scattered across the frame, with some overlapping. A white rectangular border is centered on the slide, enclosing the text.

# THANKS

## OUR TEAM

**Sparsh** - [GitHub](#) | [LinkedIn](#)

**Ashish** - [GitHub](#) | [LinkedIn](#)

**Madhurya** - [GitHub](#) | [LinkedIn](#)

**Ashwini** - [GitHub](#) | [LinkedIn](#)

**Saraiah** - [GitHub](#) | [LinkedIn](#)