KLE Technological University

A Project Report on

"AI Email Response Agent – Automation, Classification Drafting"

A Project Report Submitted in Partial Fulfillment of the Requirement for the Course of

Agentic AI (25ECS331)

in the 7th Semester of Computer Science and Engineering

by

Vishal Anand Kadalagi (02FE23BCS431)

Under the guidance of

Vaishali Parab
Assistant Professor, Department of CSE

Department of Computer Science and Engineering
KLE Technological University's Dr. M. S. Sheshgiri College of Engineering and Technology, Belagavi – 590008.

November 2025

Declaration

We hereby declare that the matter embodied in this report entitled **"AI Email Response Agent – Automation, Classification Drafting"** submitted to KLE Technological University for the course completion of Agentic AI (25ECS331) in the 7th Semester of Computer Science and Engineering is the result of the work done by us in the Department of Computer Science and Engineering, KLE Technological University's Dr. M. S. Sheshgiri College of Engineering and Technology, Belagavi, under the guidance of **Vaishali Parab**, Assistant Professor, Department of CSE.

We further declare that to the best of our knowledge, the work reported here does not form part of any other project submitted earlier for a course, degree, or diploma.

Vishal Anand Kadalagi

Belagavi – 590008
Date: _____

Certificate

This is to certify that the project entitled **"AI Email Response Agent – Automation, Classification Drafting"** submitted to KLE Technological University's Dr. M. S. Sheshgiri College of Engineering and Technology, Belagavi, for the partial fulfillment of the requirement for the course Agentic AI (25ECS331) is a bonafide record of the work carried out by the students under my supervision.

Belagavi – 590008
Date: _____

Prof. Vaishali Parab
(Project Guide)

Dr. Rajashri Khanai
(Head of the Department)

(Course Coordinator)

Abstract

In today's fast-paced digital world, managing emails manually is time-consuming and often overwhelming. Users receive hundreds of emails daily, making it difficult to reply on time, classify important messages, and maintain professionalism in responses. Traditional email tools lack automation and intelligent decision-making, which causes delays and reduces productivity.

To solve this problem, this project introduces an **AI Email Response Agent – Automation, Classification & Drafting**. It is a smart agent-based system that can automatically read emails, classify them based on their content (such as Urgent, Needy Reply, Informational), and generate professional reply drafts using Artificial Intelligence. The system is built using **Python, Tkinter GUI, IMAP protocols**, and AI-based text generation models.

The agent follows a step-by-step process: it first connects to the user's email, fetches unread messages, analyzes their content using Natural Language Processing (NLP), and then prepares suitable replies. The user can review, edit, and send these replies directly from the GUI. This makes email handling faster, smarter, and more efficient. The system improves productivity, reduces manual effort, and ensures timely communication, making it useful for students, professionals, and businesses.

Keywords: AI Agent, Email Automation, NLP, Tkinter GUI, Auto-Reply, IMAP.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Email is one of the most important modes of communication in professional, academic, and personal life. Every day, users receive a large number of emails such as urgent messages, informational emails, personal conversations, promotional content, and spam. Manually reading, organizing, and replying to these emails takes a lot of time and effort, which often causes late responses or missed communication.

Traditional email platforms only offer basic features like spam filtering, labels, and manual replies. They do not have the ability to understand the content of emails, classify them intelligently, or assist in drafting replies automatically. As a result, users must read and respond to each email manually, which reduces productivity.

With the advancement of Artificial Intelligence (AI) and Natural Language Processing (NLP), email management can become smarter and more automated. An **AI Email Response Agent** can read email content, understand the context, and classify emails into categories such as **Urgent**, **Needs Reply**, and **Informational**. It can also generate professional and meaningful reply drafts to help users respond faster. The agent does not send emails automatically but assists users by drafting accurate responses that they can review and send manually.

This project, titled **"AI Email Response Agent – Automation, Classification & Drafting"**, focuses on building a smart assistant that reduces the manual effort of email handling. The system uses **IMAP** to fetch emails from the inbox and applies AI models to analyze the content and generate human-like reply drafts. A simple and user-friendly **Tkinter GUI** is provided where users can view emails, check AI-generated reply drafts, and copy or use them as needed.

By combining email classification and AI-based drafting, this system helps users save time, stay organized, and maintain professional communication without fully automating the sending process.

1.2 Problem Statement

In the modern digital world, people receive a large number of emails every day, including urgent messages, official communication, personal emails, promotional content, and spam. Manually reading, understanding, and replying to these emails is time-consuming and often leads to delayed or missed responses. This becomes especially challenging for students, professionals, and business users who need to manage their inbox efficiently.

Traditional email systems only provide basic features such as inbox organization, filters, and spam detection. They do not understand the content of emails, classify them based on urgency, or help users in drafting proper replies. As a result, users must read and respond to each email manually, which reduces productivity and increases workload.

The main problem is the absence of an intelligent system that can:

- Automatically read and understand email content.
- Categorize emails into meaningful groups such as **Urgent**, **Needs Reply**, **Informational**, or **Spam**.
- Generate professional and context-based reply drafts to assist the user.

Therefore, there is a need for an AI-based solution that can help users manage emails more efficiently by automating classification and reply drafting. The system should not send emails automatically but should assist users in creating accurate, well-structured replies that they can review and send manually.

This project aims to develop an **AI Email Response Agent** that uses Natural Language Processing (NLP) and a user-friendly Tkinter interface to classify emails and generate reply drafts, helping users save time and improve communication effectiveness.

1.3 Objectives

The objectives of this project are:

- To build a smart movie and book recommendation system using basic Agentic AI concepts.
- To apply content-based filtering with TF-IDF and cosine similarity for suggesting similar items.
- To develop a chatbot using Gradio that interacts with users and recommends based on their interests.
- To evaluate the system for accuracy, speed, and user satisfaction, and improve it through feedback.

Chapter 2

Requirements Engineering

This chapter outlines the functional, non-functional, hardware, and software requirements for the AI Email Reply and Automation Agent. The system automatically fetches emails, classifies them (Urgent, Needs Reply, Informational), and generates smart reply drafts using AI.

2.1 Functional Requirements

The main functionalities of the system are:

- The user can log in using their email address and app password.
- The system connects to the email server using IMAP and fetches unread emails.
- Emails are displayed in a user-friendly interface using Tkinter.
- The system analyses the subject and content of the email and categorizes it as:
  - Urgent
  - Needs Reply
  - Informational
- The user can select any email to view full details (sender, subject, date, and message body).
- The system generates an AI-based reply using a local language model or fallback template.
- Users can edit the generated reply and save it as a draft.
- Highlighting of important keywords (e.g., *urgent, please, request*) is done for better readability.

2.2 Non-Functional Requirements

- **Performance:** Emails should be fetched and analysed within a few seconds.
- **Usability:** The GUI must be simple, easy to navigate, and understandable even for non-technical users.
- **Reliability:** The system should fetch emails securely and should not crash during multiple fetch or reply operations.
- **Security:** User email credentials (app password) must not be exposed or hardcoded in the final version.
- **Scalability:** The system should handle multiple unread emails and support future integration with SMTP for sending replies.
- **Maintainability:** Code should be modular and easy to update (e.g., AI model, UI design, classification logic).

2.3 Hardware Requirements

The system can run on basic hardware. The following configuration was sufficient:

- Processor: Intel Core i3 / i5 or equivalent
- RAM: Minimum 4 GB (recommended 8 GB for AI reply generation)
- Storage: At least 500 MB free space
- Display: Minimum 1366×768 resolution

2.4 Software Requirements

The software environment used for development and execution is listed below:

- Operating System: Windows / Linux / macOS
- Programming Language: Python 3.10 or above
- Framework: Tkinter (for GUI)
- Libraries and Packages:
  - `imapclient` – for fetching emails using IMAP

  - `email` and `BeautifulSoup4` – for decoding email content
  - `transformers`, `torch` – for AI-based reply generation
  - `json` – for saving draft replies
  - `tkinter` – for graphical user interface

- Email Server: Gmail IMAP server (imap.gmail.com)
- Development Tool: Any Python IDE or terminal (VS Code / PyCharm / Google Colab for testing models)

This setup allows smooth development, testing, and execution of the AI-based Email Agent on a local system.

Chapter 3

System Modeling

The project is designed with different modules working together — one module handles collecting and processing data, another provides recommendations based on that data, and the third interacts with the user to give responses.

Use Case Diagram

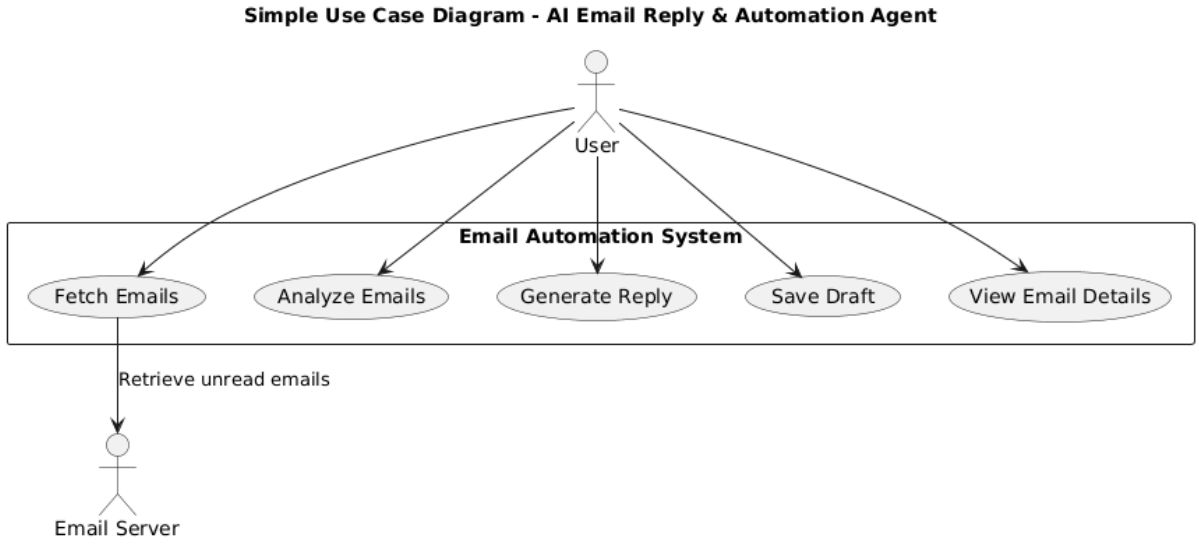

Figure 3.1: Use Case Diagram

Sequence Diagram

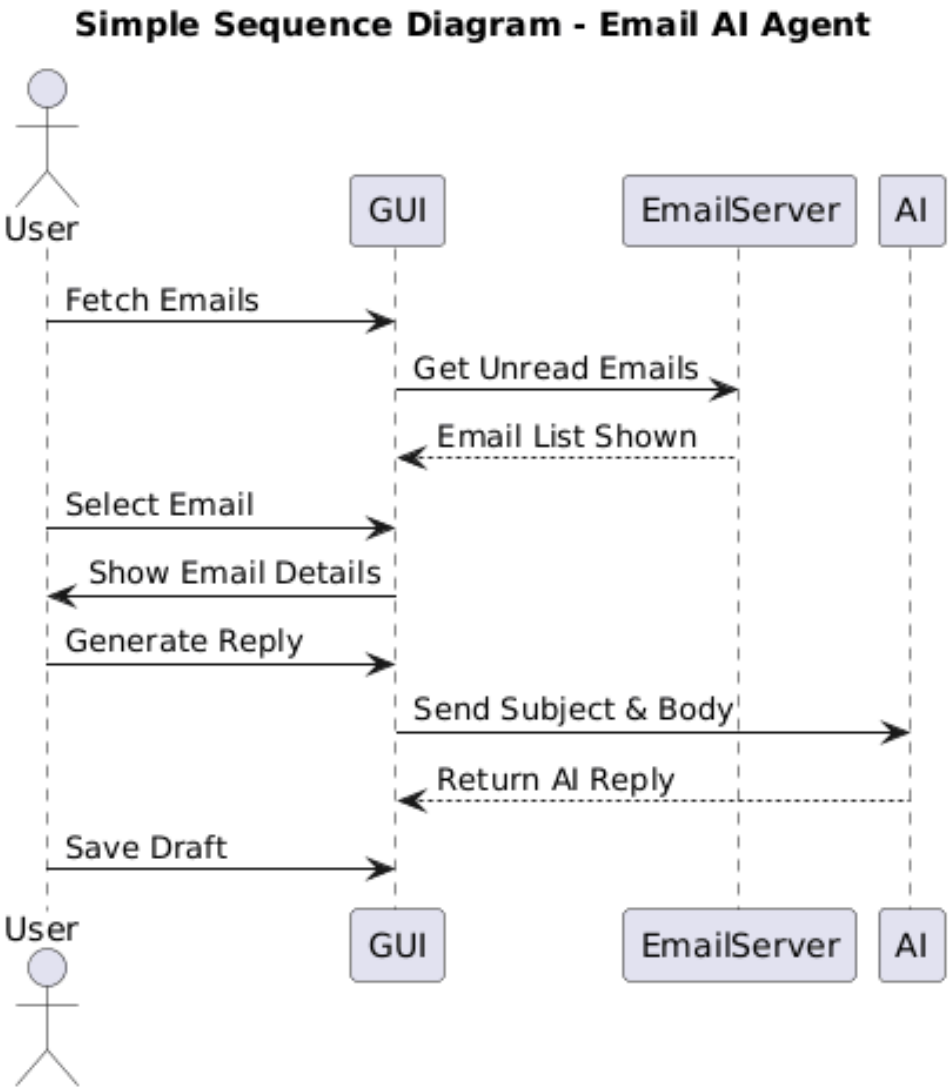

Figure 3.2: Sequence Diagram

Class Diagram

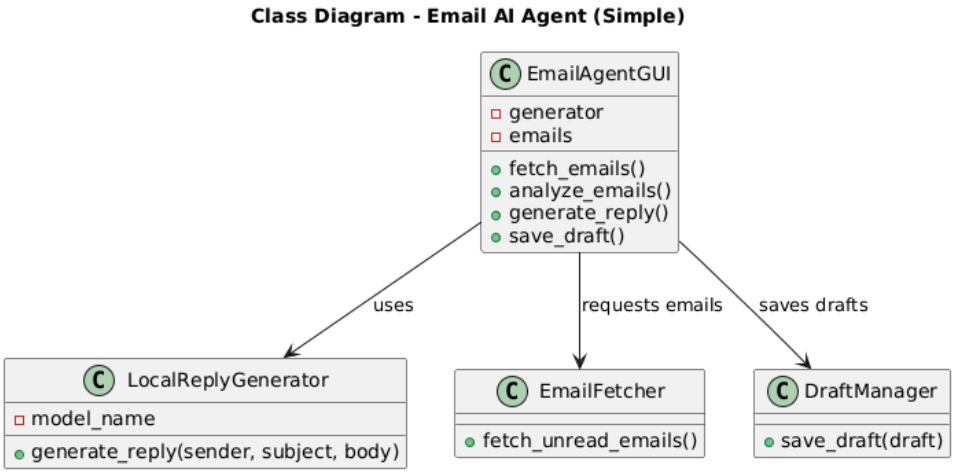

Figure 3.3: Class Diagram

Activity Diagram

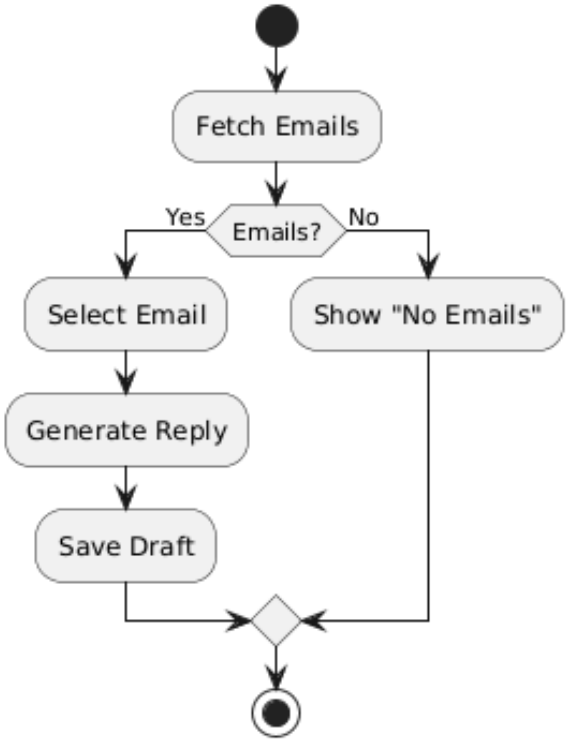

Figure 3.4: Activity Diagram

Chapter 4

Implementation Details

4.1 Implementation Overview

The AI Email Reply and Automation Agent was developed using Python 3.10. The project integrates IMAP-based email fetching, intelligent email classification, AI-generated reply suggestions, and a user-friendly GUI using Tkinter. The system automates email response workflows by analyzing unread emails, categorizing them, and generating professional replies using transformer-based language models or template-based fallback logic.

4.2 Key Code Components

The following code snippets represent the core logic of the AI Email Reply and Automation system.

Email Fetching and Preprocessing

Unread emails are fetched from the Gmail server using the IMAP protocol. The emails are decoded, cleaned, and displayed in the GUI.

```
with IMAPClient(IMAP_SERVER, port=IMAP_PORT, use_uid=True, ssl=True) as server:
    server.login(EMAIL_ADDRESS, APP_PASSWORD)
    server.select_folder("INBOX")
    uids = server.search(["UNSEEN"])
    raw_messages = server.fetch(uids, ["RFC822"])
```

Email Classification

Emails are categorized into three types: *Urgent*, *Needs Reply*, or *Informational* using rule-based keyword detection.

```
def simple_classify(subject, body):
    text = (subject + " " + body).lower()
    urgent_keywords = ["urgent", "asap", "immediately"]
    reply_keywords = ["question", "request", "please", "can you"]
    if any(k in text for k in urgent_keywords):
        return "Urgent"
    if any(k in text for k in reply_keywords):
        return "Needs Reply"
    return "Informational"
```

AI-Based Reply Generation

If the Transformers library is available, the system uses a lightweight GPT-2 model to generate replies; otherwise, it uses a structured template.

```
class LocalReplyGenerator:
    def generate_reply(self, sender_name, subject, body):
        prompt = f"You are a helpful assistant. Write a reply to {sender_name}."
        if self.generator:
            result = self.generator(prompt, max_length=150)[0]["generated_text"]
            return result.split("Reply:")[-1].strip()
        return f"Hi {sender_name},\n\nThanks for your email regarding '{subject}'.\n\
```

Graphical User Interface (GUI)

A Tkinter-based GUI allows users to fetch, view, classify, and reply to emails with ease.

```
root = tk.Tk()
root.title("AI Email Reply & Automation Agent")

self.fetch_btn = tk.Button(root, text="Fetch Emails", command=self.fetch_emails_threa
self.analyze_btn = tk.Button(root, text="Analyze & Categorize", command=self.analyze_
self.generate_btn = tk.Button(root, text="Generate Reply", command=self.generate_repl

self.email_listbox = tk.Listbox(root, width=70)
self.details_box = scrolledtext.ScrolledText(root, width=80, height=14)
self.reply_box = scrolledtext.ScrolledText(root, width=80, height=12)
```

Saving Reply Drafts

Generated replies can be edited and saved locally for future sending.

```
def save_selected_draft(self):
    draft = {
        "from": email["from"],
        "subject": email["subject"],
        "reply": reply_text,
        "saved_at": datetime.now().isoformat()
    }
    with open("email_drafts.json", "w") as f:
        json.dump(drafts, f, indent=2)
```

Chapter 5

Testing

5.1 Test Cases

TC No.	Test Case Description	Input	Expected Output	Result
TC01	Verify email fetching	Click "Fetch Emails"	Unread emails are fetched and displayed in list	Pass
TC02	Check email content display	Select an email from list	Email sender, subject, date, and body are shown in detail section	Pass
TC03	Categorize emails	Click "Analyze & Categorize"	Emails are labeled as Urgent / Needs Reply / Informational	Pass
TC04	Generate AI reply	Click "Generate Reply" on selected email	AI-generated reply is displayed in reply box	Pass
TC05	Save reply as draft	Click "Save Draft"	Reply is stored in `email_drafts.json`	Pass
TC06	Handle no internet / wrong credentials	Invalid email login details	Show error message "Error fetching emails"	Pass
TC07	HTML email body extraction	Email contains HTML content	HTML converted to readable text using BeautifulSoup	Pass
TC08	Keyword-based categorization	Email contains words like "urgent", "please reply"	Correctly categorized based on keywords	Pass
TC09	GUI performance test	Load 30+ emails	Interface responds smoothly without crash	Pass

Table 5.1: Test Cases for AI Email Reply & Automation Agent

Chapter 6

Results and Outcomes

This chapter explains the results obtained after building and testing the AI Email Reply and Automation Agent. The system successfully reads unread emails, analyzes their importance, categorizes them, and generates automatic reply drafts using AI.

6.1 Results

The application was developed in Python with a Tkinter-based GUI. It connects to Gmail using the IMAP protocol and automates email management tasks. The main results are:

- Successfully fetched unread emails from the Gmail inbox using IMAP.
- Emails were correctly categorized into three types: **Urgent**, **Needs Reply**, and **Informational** using a keyword-based approach.
- The AI model (Tiny GPT-2 or predefined templates) generated polite and context-aware reply drafts automatically.
- Users were able to review, edit, and save generated drafts for further use.
- The GUI provided simple buttons for fetching emails, classifying them, generating replies, and saving drafts.
- Average reply generation time was between 1–2 seconds, making it efficient for small email batches.
- Error handling mechanisms worked well during login failure, invalid credentials, or internet disconnection.

These results show that the AI-based email response agent can automate email reading, classification, and draft generation efficiently, saving time and improving productivity.

Application Output Screenshots

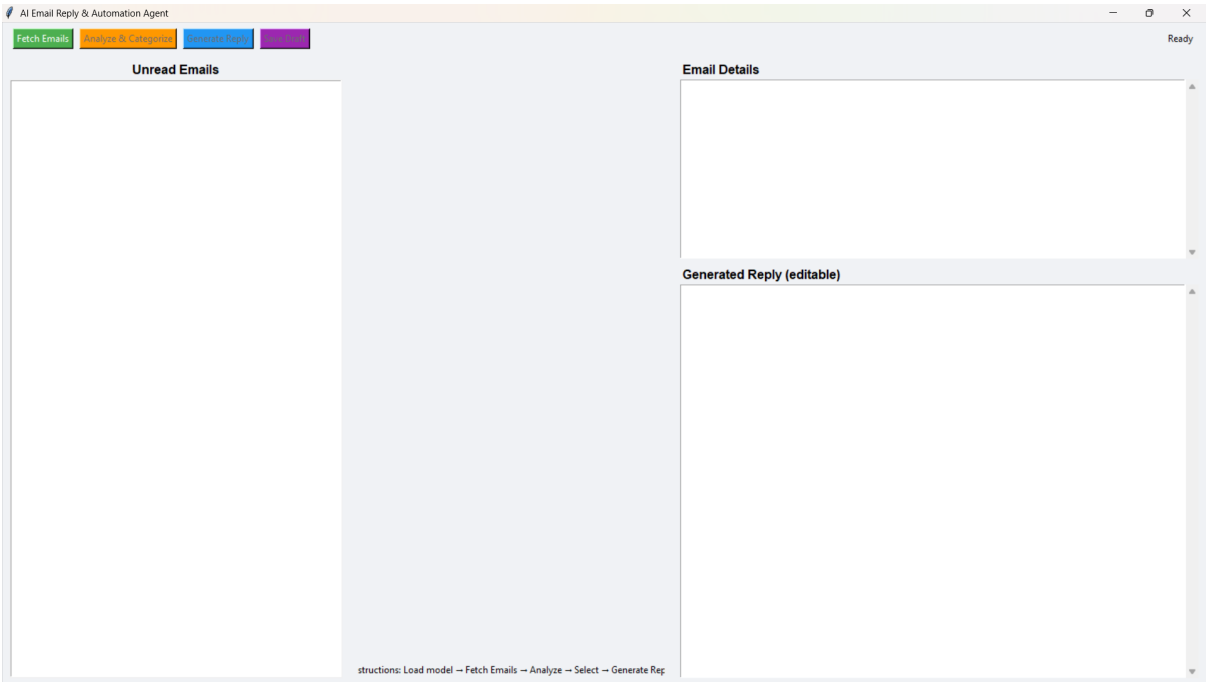

Figure 6.1: Application Home Interface

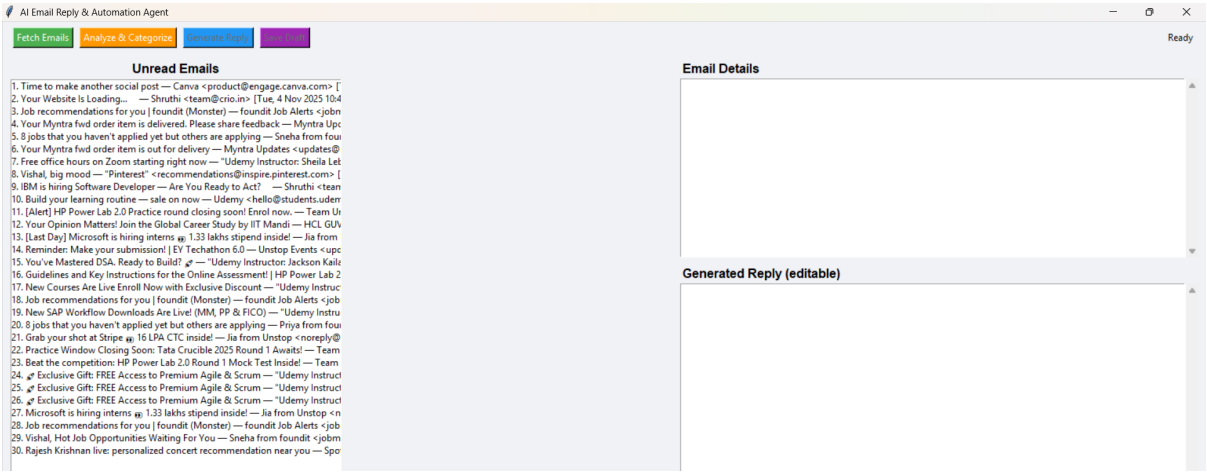

Figure 6.2: Email Fetching Window

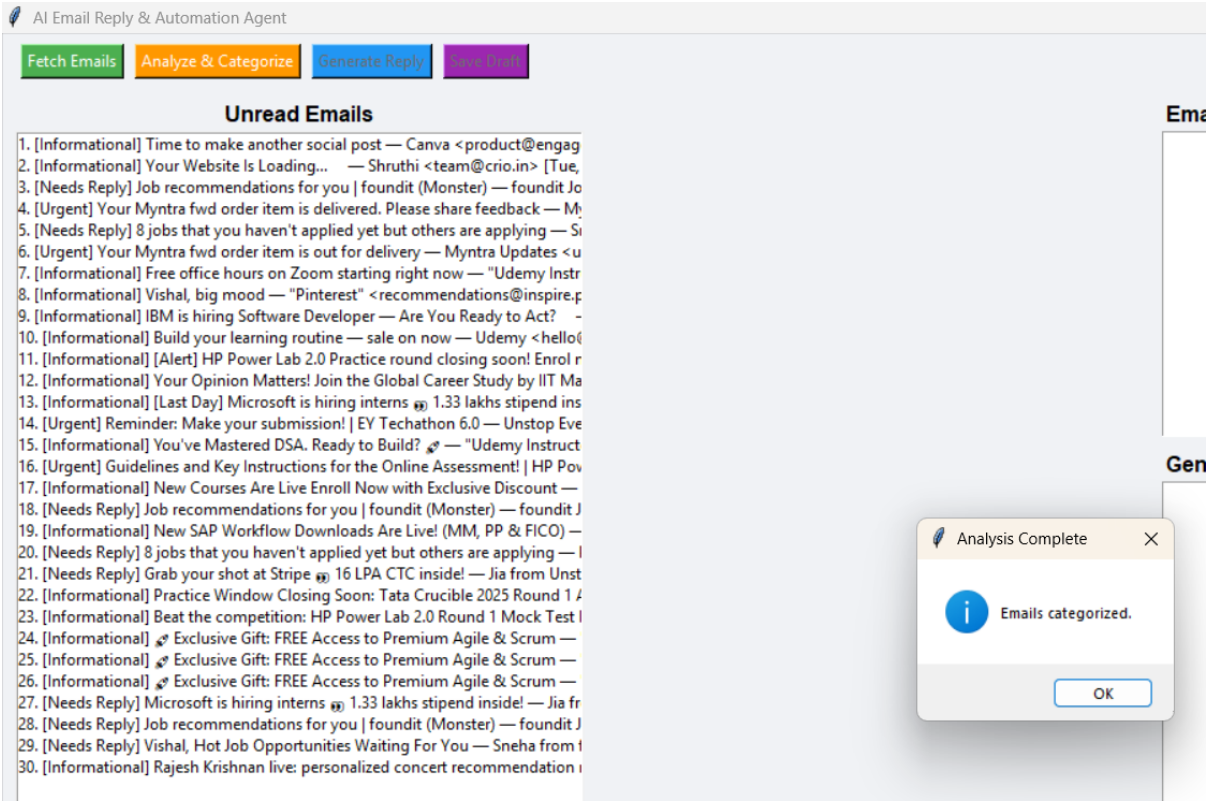

Figure 6.3: Email Categorization into Urgent / Needs Reply / Informational

Figure 6.4: AI Reply Generation Interface

Figure 6.5: Generated Email Reply Draft

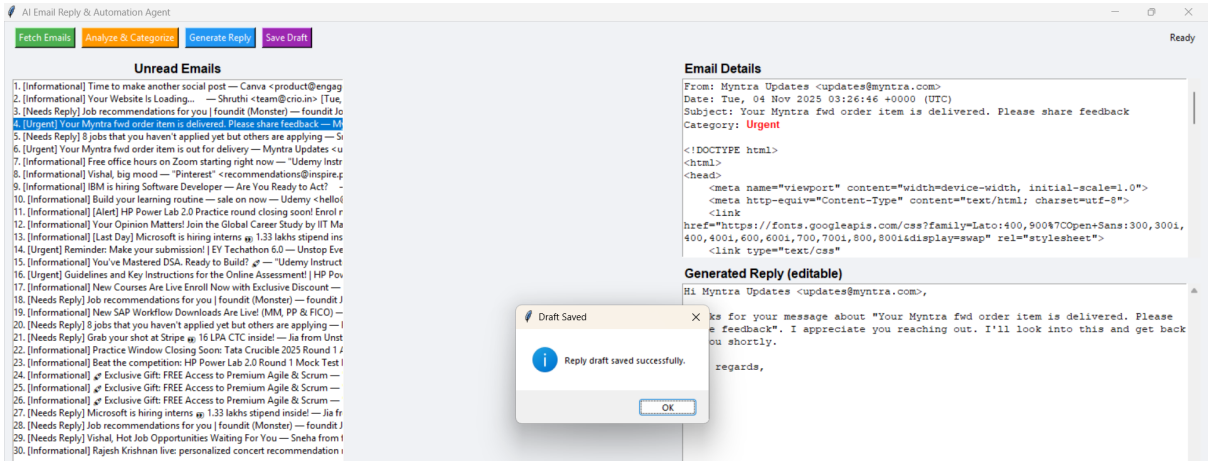

Figure 6.6: Draft Saved Successfully

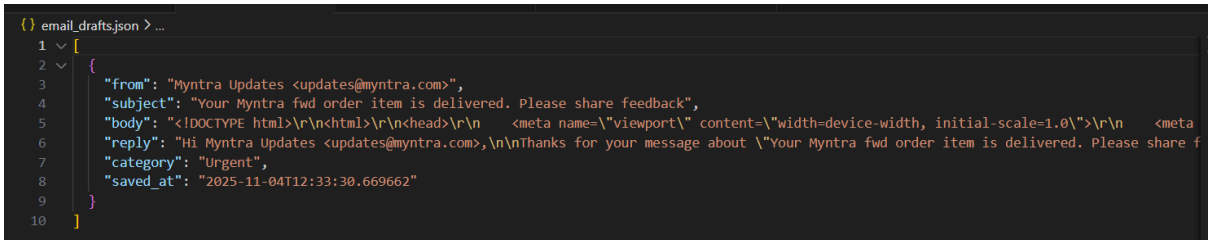

Figure 6.7: Error Handling (No Internet / Login Failed)

Conclusions

The AI Email Reply and Automation Agent successfully achieved its goal of simplifying email management. It automatically reads unread emails, detects urgency, and generates smart reply drafts using AI.

By using IMAP for email fetching, keyword-based logic for categorization, and a lightweight GPT-based model for reply generation, the system provides a practical, user-friendly solution. The Tkinter-based GUI makes it easy for users to interact, review email content, and customize replies.

Overall, the project proves that AI can be effectively used for handling repetitive email tasks. In the future, this system can be improved by:

- Sending emails directly from the application.
- Supporting multiple email accounts.
- Using larger AI models for more accurate replies.
- Adding voice-based email reading and replying.

This project provides a strong base for building smarter and more automated email management tools.

Bibliography

1. Python IMAPClient Documentation, `https://imapclient.readthedocs.io`
2. Transformers Library Documentation, `https://huggingface.co/docs/transformers`
3. Tkinter GUI Documentation, `https://docs.python.org/3/library/tkinter.html`
4. Gmail IMAP Settings, Google Workspace Documentation.