# Ahmed Samir

# 8120

## Introduction

The task was to create a program that sends emails using the SMTP protocol and receives emails using the IMAP protocol. The application was implemented using three scripts: SendEmail.py for sending emails, receive_email.py for receiving emails, and App.py for the GUI.

Initially, **mail.tm** was intended as the testing server, but due to persistent DNS resolution issues, **Gmail** was used instead. This report details the code, usage instructions, dependencies, testing process, and results, fulfilling all assignment requirements.

---

## Code Description

The application is split into three Python scripts, each with a specific role, ensuring modularity and readability. Below is a detailed breakdown:

### SendEmail.py

- **Purpose**: Handles email sending via the SMTP protocol.
- **Implementation**:
  - Uses smtplib to establish a TCP connection to smtp.gmail.com on port 587 with TLS encryption.
  - Constructs emails using email.mime.multipart.MIMEMultipart for headers (From, To, Subject) and MIMEText for the body.
  - Parameters: sender_email, sender_password, recipient_email, subject, body, output_text.
  - Steps: Resolves the SMTP server hostname, connects, enables TLS, logs in, sends the email, and closes the connection.
- **Error Handling**:

    - Catches socket.gaierror for DNS issues, smtplib.SMTPAuthenticationError for login failures, and general exceptions.
    - Outputs status or errors to the output_text widget (GUI) or console (standalone).

**ReceiveEmail.py**

- **Purpose**: Retrieves the latest email from the inbox using IMAP.
- **Implementation**:
    - Uses imaplib.IMAP4_SSL to connect to imap.gmail.com on port 993 with SSL.
    - Parses emails with email.message_from_bytes and decode_header for subject decoding.
    - Parameters: email_user, email_pass, output_text.
    - Steps: Connects securely, logs in, selects the inbox, searches for all emails, fetches the latest one (highest ID), and extracts subject, sender, and body.
- **Error Handling**:
    - Handles connection failures, login errors, and fetch issues with descriptive messages.
    - Checks for empty inboxes and reports "No emails found."
- **Comments**: Explains key actions, e.g., "Fetch the latest email by ID."

**App.py**

- **Purpose**: Provides a user-friendly GUI integrating sending and receiving functionalities.
- **Implementation**:
    - Built with tkinter, featuring three windows:
        1. **Choice Window**: Asks "Send" or "Receive" (300x150, non-resizable).
        2. **Send Window**: Full interface (600x700) with fields for sender email, app password, recipient, subject, and body, plus a "Send Email" button.

3. **Receive Window**: Minimal interface (400x500) with email and password fields, plus a “Receive Latest Email” button.
    - Imports send_email and receive_email from their scripts.
    - Uses scrolledtext for real-time output and messagebox for input validation errors.
- **Design**:
    - Light gray background (#f0f0f0), Helvetica fonts for labels/buttons, Courier for output.
    - Bold buttons and padded layout for a professional look.
- **Comments**: Describes GUI structure and event handling.

### Deviation from Requirements

- The lab suggested mail.tm for testing, but DNS resolution failures **([Errno 11001] getaddrinfo failed)** with smtp.mail.tm and smtp.temp-mail.io which made me switch to Gmail.

---

## How to Use the Application

### Installation

1. **Scripts**: Download SendEmail.py, ReceiveEmail.py, and App.py into a single directory
2. **Gmail Setup**:
    - Create a Gmail account or use an existing one (e.g., your_email@gmail.com).
    - Enable 2-Step Verification in Google Account settings.
    - Generate an App Password at myaccount.google.com/apppasswords (16-character code, e.g., xxxx xxxx xxxx xxxx).

### Running the Application

1. **Launch GUI**: Run python App.py.

2. **Using the GUI**:
    - **Choice Window**:
        - Click “Send Email” to open the send window.
        - Click “Receive Email” to open the receive window.
    - **Send Window**:
        - Enter sender email (e.g., ahmedsamirelboridy@gmail.com), App Password, recipient email (e.g., samarasamir4234@gmail.com), subject, and body.
        - Click “Send Email” to send; view progress in the output area (e.g., “Email sent successfully”).
    - **Receive Window**:
        - Enter email address and App Password.
        - Click “Receive Latest Email” to fetch the latest email; see subject, sender, and body in the output area.

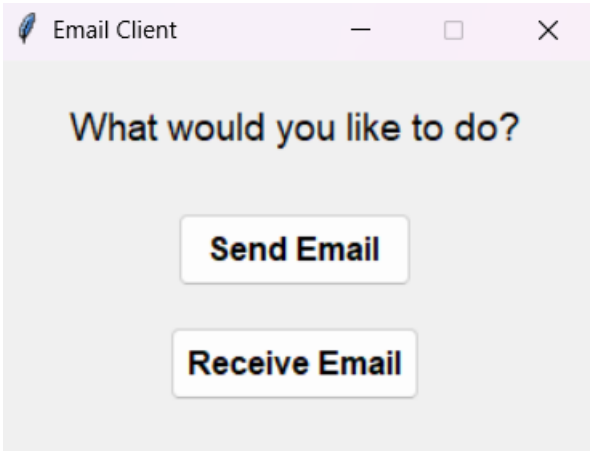
---

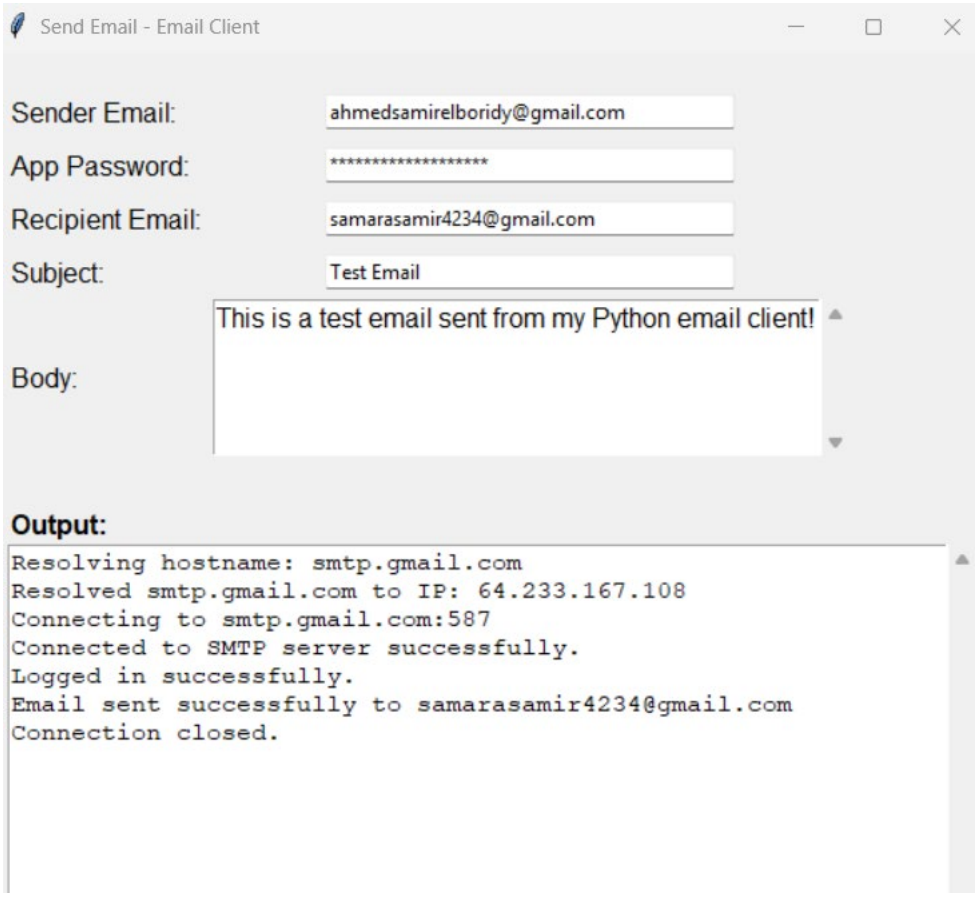
## Dependencies

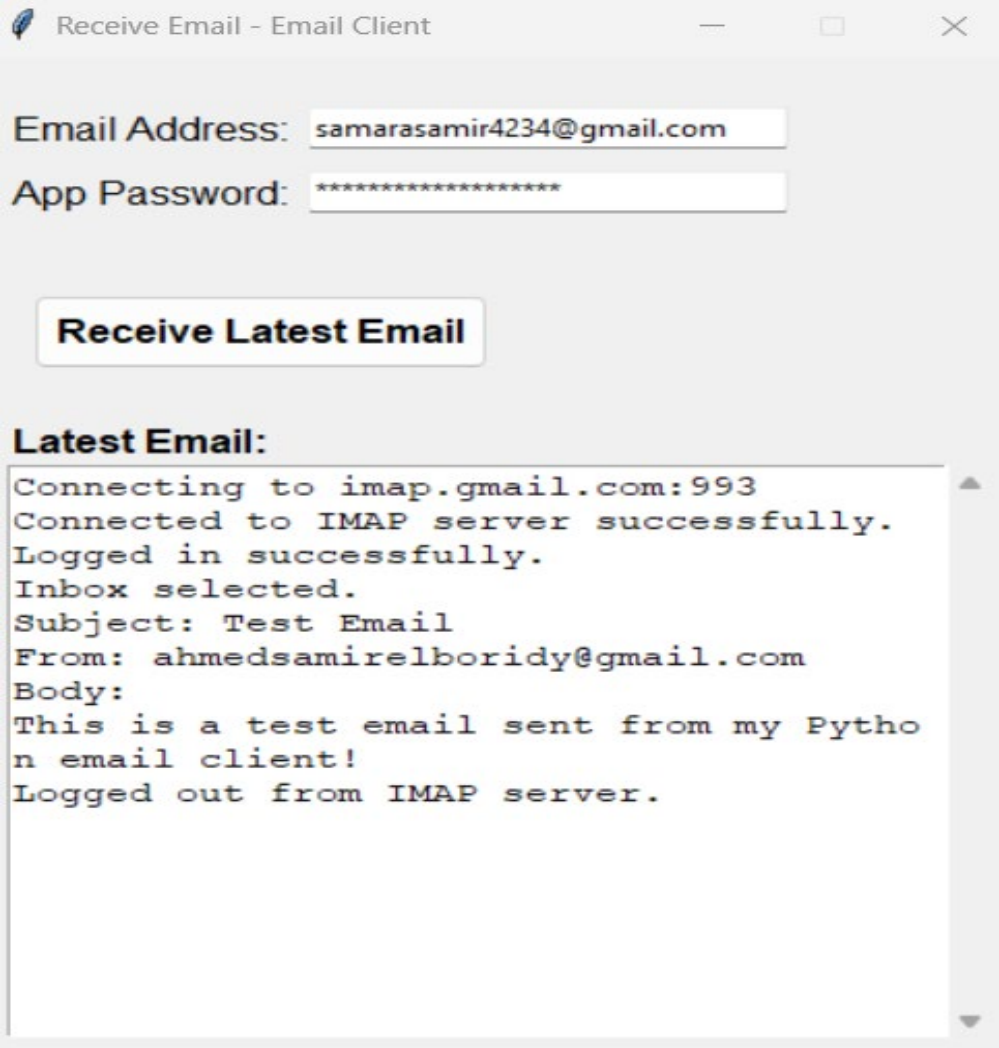
The application relies on standard Python libraries, requiring no external installations:

- **smtplib**: Sends emails via SMTP.
- **imaplib**: Receives emails via IMAP.
- **email**: Constructs and parses email messages.
- **socket**: Handles DNS resolution for error checking.
- **tkinter**: Provides GUI functionality (includes ttk, scrolledtext, messagebox).
- **External Service**: Gmail account with an App Password for authentication. No additional pip packages are needed, ensuring easy setup.

## Some testing outputs results :






---

## Conclusion

The email client fulfills all core requirements, sending and receiving emails with SMTP and IMAP, robust error handling, and clean code structure.