# Python Automation : Email Reminder Scheduler

## Objective

Create a Flask-based Python web application that allows users to schedule and send automated email reminders using Gmail SMTP.

## Tools & Libraries

* Python 3
* Flask
* smtplib
* email.mime (for composing email)
* schedule
* threading

## Steps Implemented

1. Enabled Gmail 2-Step Verification and generated an App Password for SMTP authentication.

2. Developed a Flask web application (email\_reminder\_app.py) to collect sender, receiver, message, and time from the user.

3. Created interactive HTML pages (index.html and success.html) for input and confirmation.

4. Integrated Flask with smtplib to send scheduled emails automatically at user-defined times.

5. Tested the web application successfully — email reminder received in the inbox at the scheduled time.

## Code Implementation

from flask import Flask, render\_template, request

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

import schedule

import time

import threading

app = Flask(\_\_name\_\_)

# Function to send email

def send\_email(sender\_email, receiver\_email, app\_password, subject, body):

    message = MIMEMultipart()

    message["From"] = sender\_email

    message["To"] = receiver\_email

    message["Subject"] = subject

    message.attach(MIMEText(body, "plain"))

    try:

        with smtplib.SMTP("smtp.gmail.com", 587) as server:

            server.starttls()

            server.login(sender\_email, app\_password)

            server.send\_message(message)

        print("✅ Email sent successfully!")

    except Exception as e:

        print("❌ Error:", e)

# Run schedule in background

def run\_schedule():

    while True:

        schedule.run\_pending()

        time.sleep(30)

@app.route("/", methods=["GET", "POST"])

def index():

    if request.method == "POST":

        sender\_email = request.form["sender"]

        app\_password = request.form["password"]

        receiver\_email = request.form["receiver"]

        subject = request.form["subject"]

        body = request.form["body"]

        time\_input = request.form["time"]

        schedule.clear()

        schedule.every().day.at(time\_input).do(

            send\_email, sender\_email, receiver\_email, app\_password, subject, body)

        return render\_template(

            "success.html",

            sender=sender\_email,

            receiver=receiver\_email,

            subject=subject,

            body=body,

            time\_input=time\_input,

        )

    return render\_template("index.html")

if \_\_name\_\_ == "\_\_main\_\_":

    threading.Thread(target=run\_schedule).start()

    app.run(debug=True)

## HTML Templates

Two HTML files were created for frontend interaction: index.html and success.html

**Index.html**

<!DOCTYPE html>

<html>

<head>

    <title>Email Reminder</title>

    <style>

        body { font-family: Arial; background: #f4f4f9; padding: 40px; }

        form { background: white; padding: 20px; border-radius: 10px; max-width: 400px; margin: auto; box-shadow: 0 0 10px rgba(0,0,0,0.1); }

        input, textarea { width: 100%; padding: 10px; margin: 8px 0; border-radius: 5px; border: 1px solid #ccc; }

        button { background: #007BFF; color: white; border: none; padding: 10px 15px; border-radius: 5px; cursor: pointer; width: 100%; }

        button:hover { background: #0056b3; }

        h2 { text-align: center; color: #333; }

    </style>

</head>

<body>

    <h2>📧 Email Reminder Scheduler</h2>

    <form method="POST">

        <label>Sender Email:</label>

        <input type="email" name="sender" required>

        <label>App Password:</label>

        <input type="password" name="password" required>

        <label>Receiver Email:</label>

        <input type="email" name="receiver" required>

        <label>Subject:</label>

        <input type="text" name="subject" required>

        <label>Message:</label>

        <textarea name="body" rows="4" required></textarea>

        <label>Time (24hr format):</label>

        <input type="text" name="time" required>

        <button type="submit">Schedule Email</button>

    </form>

</body>

</html>

## Success.html

<!DOCTYPE html>

<html>

<head>

    <title>Reminder Scheduled</title>

    <style>

        body { font-family: Arial; background: #eaf4ff; padding: 40px; text-align: center; }

        .box { background: white; padding: 25px; border-radius: 10px; display: inline-block; box-shadow: 0 0 10px rgba(0,0,0,0.1); }

        h2 { color: #007BFF; }

        p { color: #333; font-size: 16px; }

        span { font-weight: bold; color: #555; }

        a { display: inline-block; margin-top: 15px; text-decoration: none; background: #007BFF; color: white; padding: 8px 12px; border-radius: 5px; }

        a:hover { background: #0056b3; }

    </style>

</head>

<body>

    <div class="box">

        <h2>✅ Email Reminder Scheduled Successfully!</h2>

        <p><span>From:</span> {{ sender }}</p>

        <p><span>To:</span> {{ receiver }}</p>

        <p><span>Subject:</span> {{ subject }}</p>

        <p><span>Message:</span> {{ body }}</p>

        <p><span>Scheduled Time:</span> {{ time\_input }}</p>

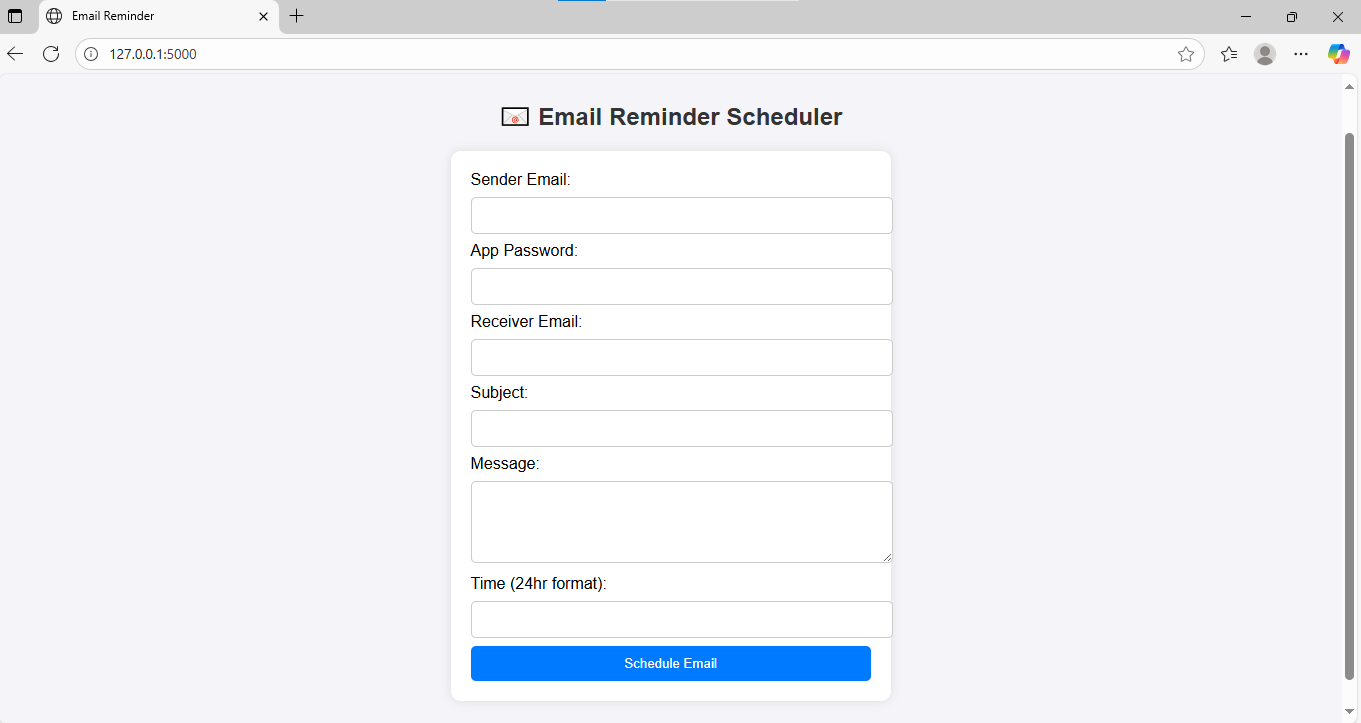
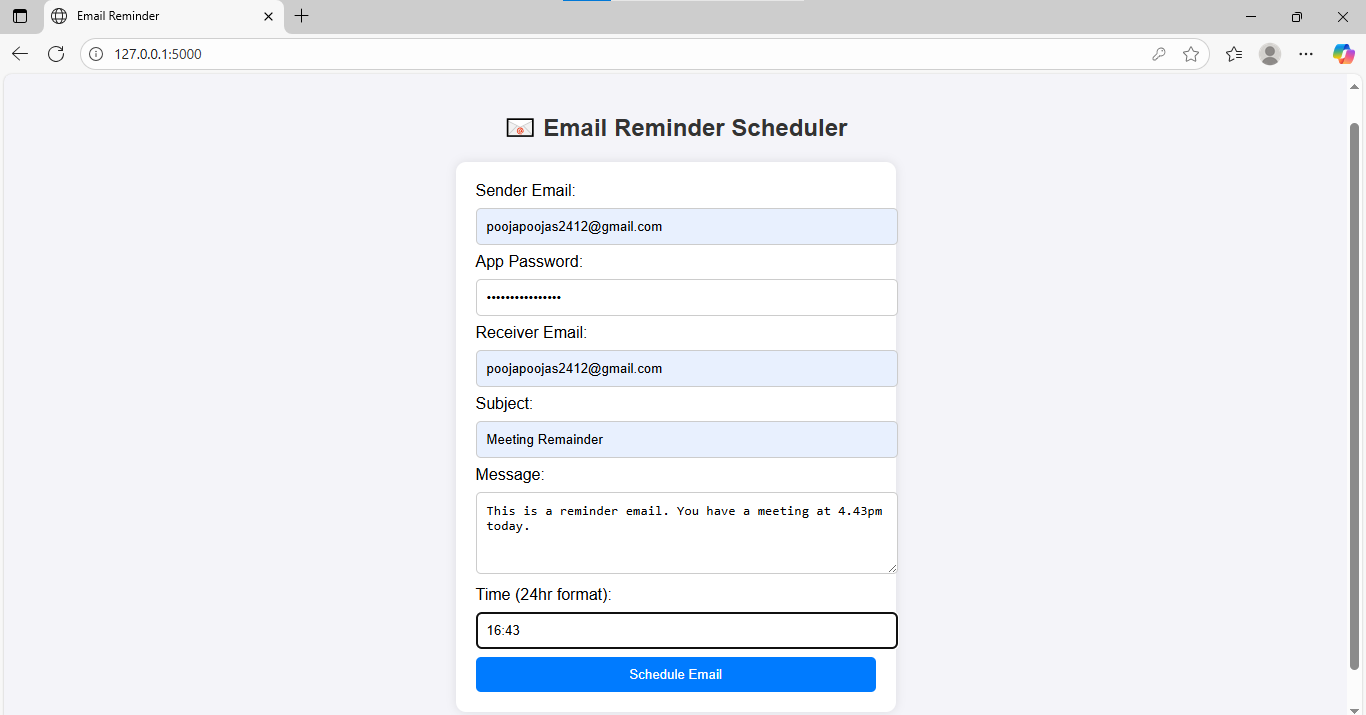
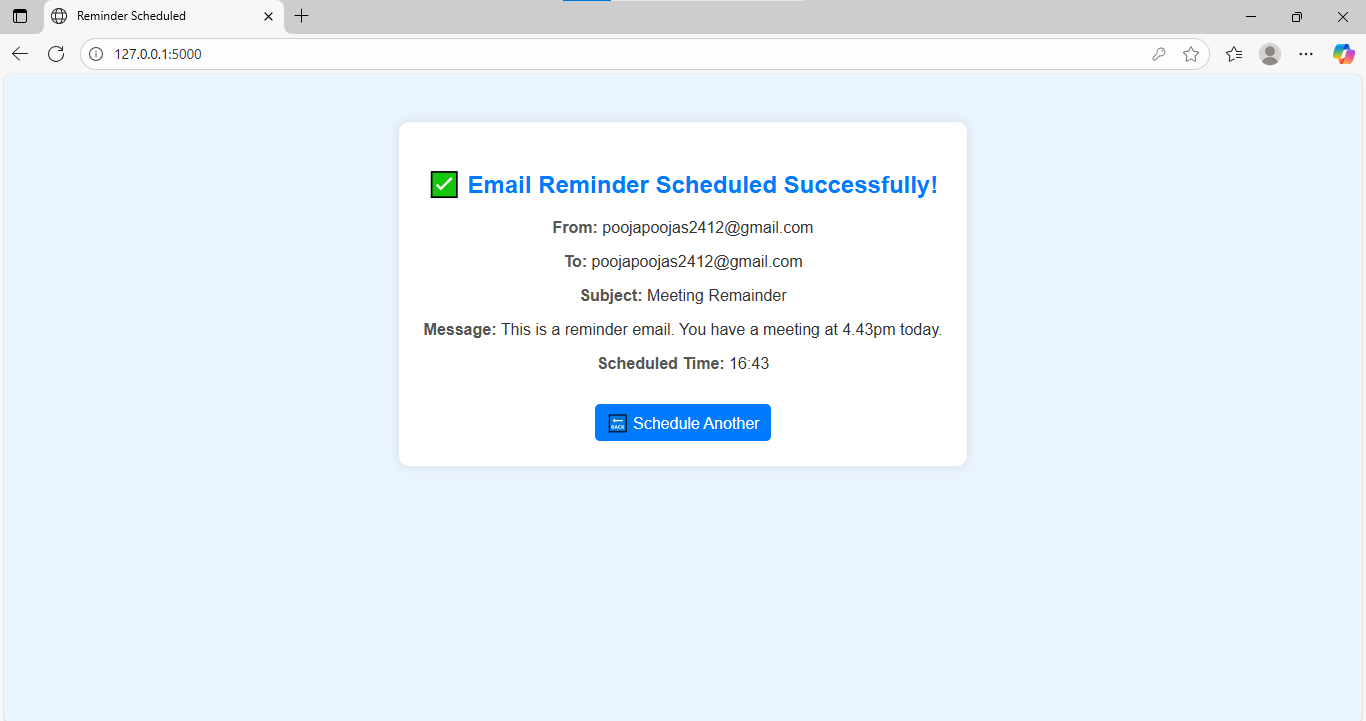
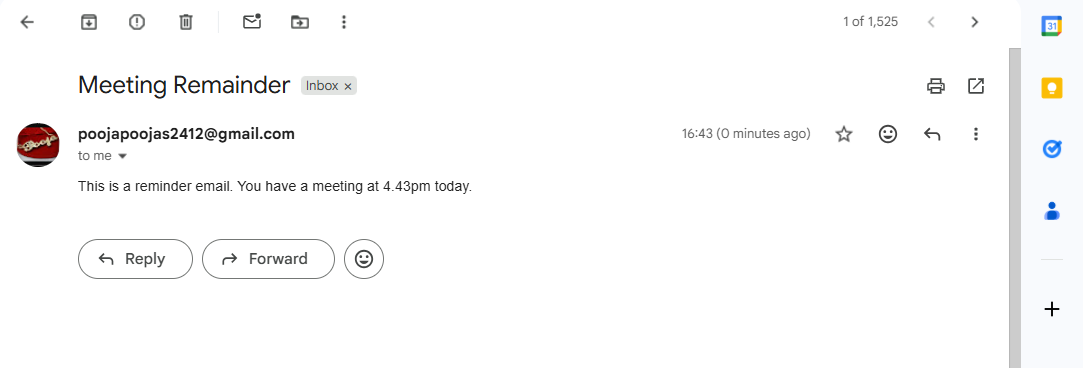
        <a href="/">🔙 Schedule Another</a>

    </div>

</body>

</html>

## Output ScreenShot

**Advantages**

* Interactive and user-friendly web interface using Flask.
* Automated email sending and scheduling.
* Eliminates manual configuration for sender and receiver in code.
* Demonstrates integration of Python backend with web frontend.

## Future Enhancements

* Add database support to store scheduled reminders.
* Allow multiple recipients and recurring reminders.
* Add email templates with HTML formatting.

## Conclusion

The Flask-based Email Reminder Scheduler successfully integrates a Python backend with a web interface to automate email reminders.