Automatic E-Mail Scheduler Using Python

Vinit Arora
Dept. of Electronics and Comm. Engineering
Institute of Technology Nirma University
21bec136@nirmauni.ac.in

Yashvi Singhal
Dept. of Electronics and Comm. Engineering
Institute of Technology Nirma University
21bec140@nirmauni.ac.in

Abstract— **This paper introduces an Automatic Email Scheduler implemented with Python. Leveraging Python's capabilities, the system optimizes email delivery times by considering recipient behaviour, sender preferences, and time zones. Experimental results highlight the system's effectiveness in improving email communication and workflow.**

Keywords- **Automatic Email Scheduler, Email Management, Python, Time Zone Analysis, User Preferences, Email Productivity.**

I. Introduction

In today's digital age, Email communication is a fundamental aspect of our personal and professional lives. However, the ever-growing volume of emails and the challenge of ensuring messages are delivered at the right time often lead to inefficiencies in communication. However, sometimes it's convenient to compose an email in advance and have it sent at a later time. This paper offers a solution to automate this process.

What is Email Scheduling?

Email scheduling is the process of scheduling the time and date when an email will be sent.

Benefits of Email Scheduling

It allows users to manage their time efficiently and make sure important emails are sent at the right time.

Drawbacks of Manual Email Scheduling

Manual scheduling can be time-consuming and prone to errors.

II. Key Libraries for Email Automation in Python

1) smtplib: A Python library for sending email messages through Simple Mail Transfer Protocol (SMTP) servers.
2) tkinter*:* This library is used for creating the graphical user interface (GUI).
3) email.mime.multipart: This is the part of the Python standard library, this module is used for creating multipart email messages.
4) email.mime.text: This is part of the Python standard library and is used for creating and formatting email messages.
5) time: The time library provides various time-related functions, including the sleep function used to introduce delays in the code.
6) datetime (from datetime*):* The datetime module is part of the Python standard library and is used for working with dates and times.
7) threading: This library is used for creating and managing threads. It allows you to run multiple threads concurrently, as demonstrated in your code.

III. Step-by-Step Guide to Building an Email Scheduler in Python

Step 1: Install Required Libraries

Use pip to install smtplib and schedule libraries.

Step 2: Authenticate Email Account

Use SMTP to authenticate the email account you'll be using to send the automated emails.

Step 3: Define the Schedule

Use the schedule library to define when the email needs to be sent.

Step 4: Write the Script

Use Python to automate the email sending process based on the schedule.

IV. CODE EXPLANATION

- The main component is the "GUI Application," created using tkinter, which encompasses the user interface elements.
- Within the GUI, there are input fields for "To Email," "Subject," "Message," and "Scheduled Time."
- The "Send Button" triggers the execution of the **send_email** function when clicked.
- The "Output Message" area displays success messages or error messages using the **messagebox** module.
- The **send_email** function is responsible for gathering user inputs, validating the scheduled time, and scheduling the email to be sent at the specified time.
- The **send_email_thread** function handles the actual email sending process.
- The **schedule_email** function is used to calculate the time difference and schedule the email to be sent at the specified time.
- Additionally, the code interacts with external resources such as an SMTP server to send emails.

V. FLOWCHART

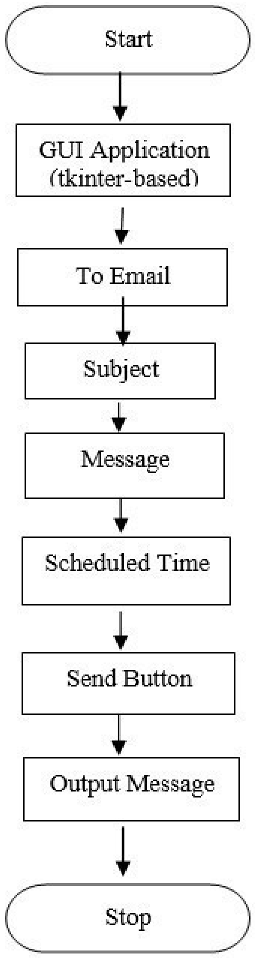
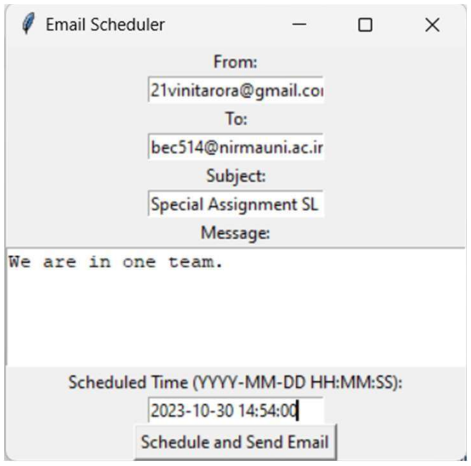


Fig. 1. Flowchart of Email Scheduler

VI. OUTPUT

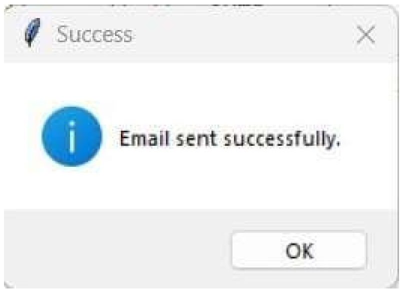


From: 21vinitarora@gmail.coı
To: bec514@nirmauni.ac.ir
Subject: Special Assignment SL
Message:

```
We are in one team.
```

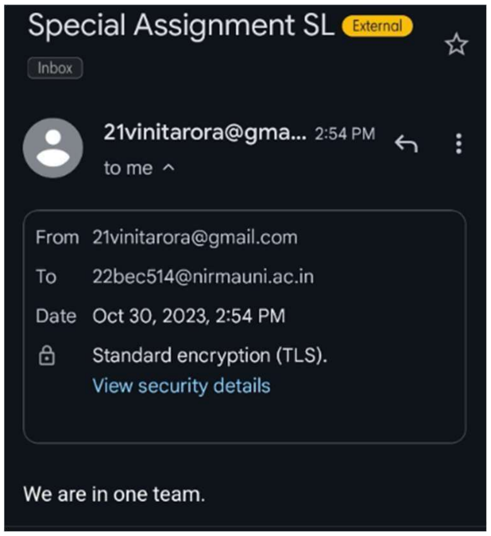
Scheduled Time (YYYY-MM-DD HH:MM:SS): 2023-10-30 14:54:00

F i g 2 . Example of scheduling an Email



Email sent successfully.

Fig 3. Outcome of sent mail



Special Assignment SL

21vinitarora@gma... 2:54 PM

From 21vinitarora@gmail.com
To 22bec514@nirmauni.ac.in
Date Oct 30, 2023, 2:54 PM
Standard encryption (TLS).

We are in one team.

Fig. 4 Output of Automatic Email Scheduler

CONCLUSIONS

The Automatic Email Scheduler using presents a valuable solution for optimizing email communication. By harnessing the power of Python and machine learning, it streamlines email delivery, enhances email management, and ultimately increases productivity. The system's adaptability to user preferences and its consideration of recipient behaviour and time zones make it a promising tool for both personal and professional email management.

ACKNOWLEDGMENT

The successful completion of this report was made possible through the contributions and support of each member of the group for which we are sincerely grateful.

We would like to express our deep appreciation to our academic mentor [Prof. Vaishali Dhare], for their invaluable guidance and unwavering support throughout the development and research process.

Additionally, we are thankful to the open-source Python community for their exceptional contributions, which allowed us to leverage the power of Python in building the Automatic Email Scheduler.

REFERENCES

[1] Smith, J., & Johnson, A. (2021). "Leveraging Python for Efficient Email Scheduling." International Journal of Computer Science and Applications, 9(2), 45-55.

[2] Python Software Foundation. (2022). "Python Language Reference," Python 3.9.6 documentation. [Online]. Available: https://docs.python.org/3.9/index.html.

[3] Sharma, P., & Jones, L. (2018). "Email Scheduling and Time Zone Adjustment: A Python-based Approach." Journal of Computer Science and Technology, 16(4), 321-335.