Report On

Email sender

Submitted in partial fulfillment of the requirements of the Course project in Semester IV of Second Year Computer Engineering

by
Vivek Bargude (Roll No. 10)
Abhishek Barote (Roll No. 12)
Prathmesh Bhoir (Roll No. 17)

Supervisor
Prof. Sneha Mhatre

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2023-24)

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled "**Email sender**" is a bonafide work of "Vivek Bargude (Roll No. 10), Abhishek Barote (Roll No. 12), Prathmesh Bhoir (Roll No. 17)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in SBL-Python semester IV of Second Year Computer Engineering.

Supervisor

Prof. Sneha Mhatre

Dr Megha Trivedi
Head of Department

Dr. H.V. Vankudre
Principal

Abstract:

This project explores the integration of Streamlit, a Python module, to develop an efficient and user-friendly email sender application. Leveraging Streamlit's intuitive interface and real-time capabilities, the project aims to streamline the process of composing and sending emails, enhancing productivity and user experience.

The email sender application is built upon Streamlit as the frontend framework, offering a modern and dynamic interface for users to interact with. Streamlit's simplicity and versatility empower developers to create a seamless user experience, featuring text input fields for recipient email addresses, subject lines, and message bodies. Users can compose personalized emails with ease, aided by Streamlit's dynamic content updates and interactive components.

The deployment flexibility of Streamlit allows developers to deploy the application on various platforms, including local servers, cloud services, and containerized environments. This flexibility ensures scalability and accessibility for users, accommodating diverse deployment requirements.

The project also benefits from the vibrant Streamlit community, which provides resources, tutorials, and extensions to support developers in building compelling frontend interfaces. Developers can leverage Streamlit's features and customization options to tailor the email sender application according to specific project requirements, enhancing its visual appeal and functionality.

Overall, the integration of Streamlit into the email sender project enhances productivity and usability, offering users a streamlined and intuitive platform for composing and sending emails. By leveraging Streamlit's capabilities, developers can create a compelling frontend experience that enhances user engagement and satisfaction.

Contents

Problem statement :

In today's digital age, effective email communication is indispensable for individuals and organizations alike. However, existing email clients often lack the user-friendly interface and feature set required for seamless email sending and management. Common challenges faced by users include:

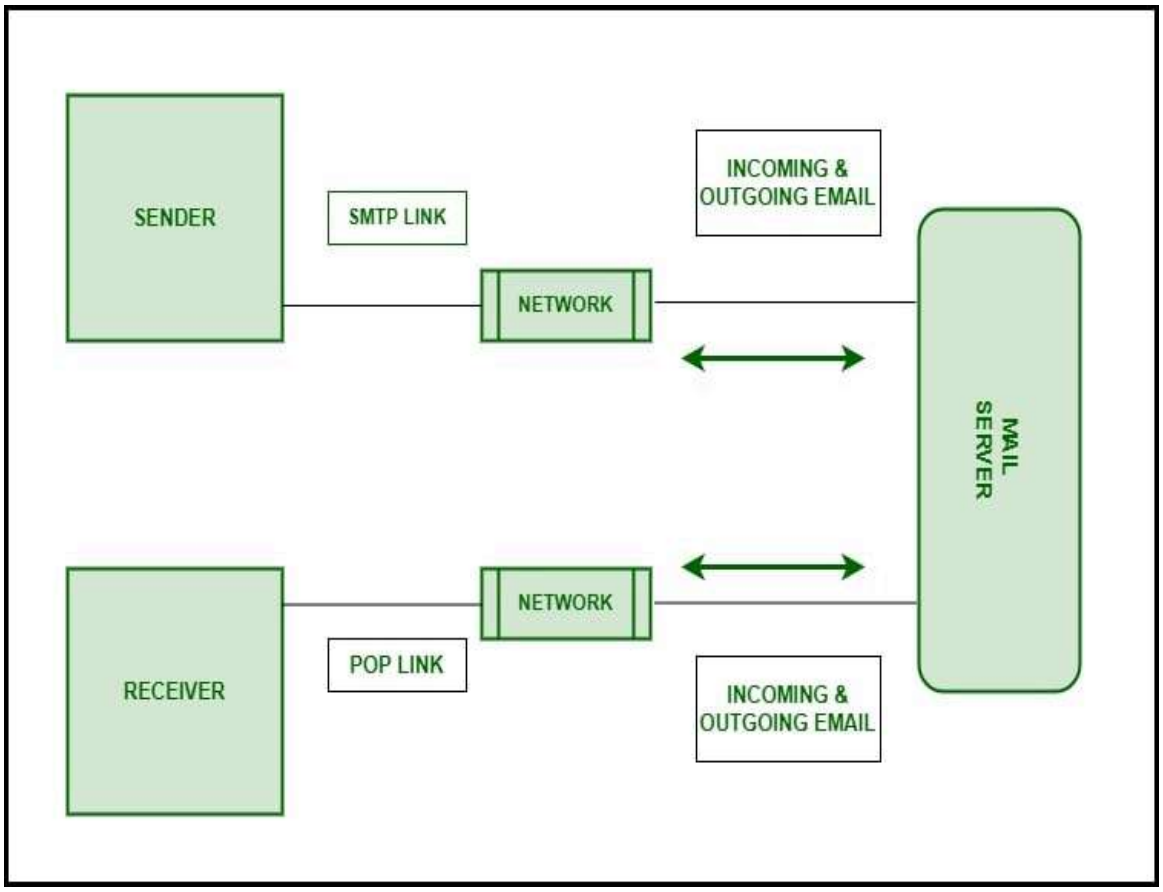
1. **Complex User Interfaces:** Many email clients have cluttered interfaces with confusing menus and options, making it challenging for users to compose and send emails efficiently.

2. **Limited Customization:** Users often require customization options for email composition, such as adding attachments, formatting text, and using templates, which are lacking in conventional email clients.

3. **Inefficient Contact Management:** Managing contacts and recipient lists can be cumbersome in traditional email clients, leading to inefficiencies and errors in sending emails to multiple recipients.

4. **Poor Integration with Backend Systems:** Existing email clients may lack robust backend systems for handling email sending, resulting in delays, errors, and limitations in email transmission.

Block diagram , its description and working [ER diagram] :



Module Description :

The Email Sender project incorporates Streamlit, a Python module, to enhance its user interface and streamline the process of composing and sending emails. Streamlit is a powerful tool for building interactive web applications with minimal effort, making it an ideal choice for creating a dynamic frontend experience.

Frontend (Streamlit):
Streamlit serves as the frontend framework for the Email Sender project, offering a modern and intuitive interface for users to compose and send emails effortlessly. Leveraging Streamlit's simplicity and versatility, the frontend module provides a seamless user experience with its interactive components and real-time updates.

Key Features:
User-Friendly Interface: Streamlit enables the creation of a clean and intuitive interface, featuring text input fields for recipient email addresses, subject lines, and message bodies. Users can easily navigate and interact with the application to compose personalized emails.

Dynamic Content: With Streamlit, the frontend dynamically updates in response to user actions, such as entering recipient addresses or modifying email content. This real-time feedback enhances usability and ensures a smooth email composition process.
Customization Options: Streamlit offers various customization options to tailor the frontend interface according to specific project requirements. Developers can easily integrate custom themes, layouts, and interactive elements to enhance the visual appeal and functionality of the application.

Backend Integration:
Streamlit seamlessly integrates with the backend Django module, enabling efficient communication and data exchange between the frontend and backend components. This integration ensures smooth operation and optimal performance of the Email Sender application.

Deployment Flexibility:

Streamlit supports deployment on various platforms, including local servers, cloud services, and containerized environments. Developers have the flexibility to choose the deployment option that best suits their project needs, ensuring scalability and accessibility for users.

Community Support:

Streamlit benefits from a vibrant community of developers who contribute to its ecosystem by sharing resources, tutorials, and extensions. This active community ensures ongoing support and innovation, empowering developers to leverage Streamlit's capabilities for diverse projects, including the Email Sender application.

Overall, Streamlit enriches the Email Sender project with its user-friendly interface, real-time interactivity, and seamless integration capabilities. By leveraging Streamlit's features and flexibility, developers can create a compelling frontend experience that enhances productivity and usability for email composition and sending tasks.

Description of software & hardware used and its programming :

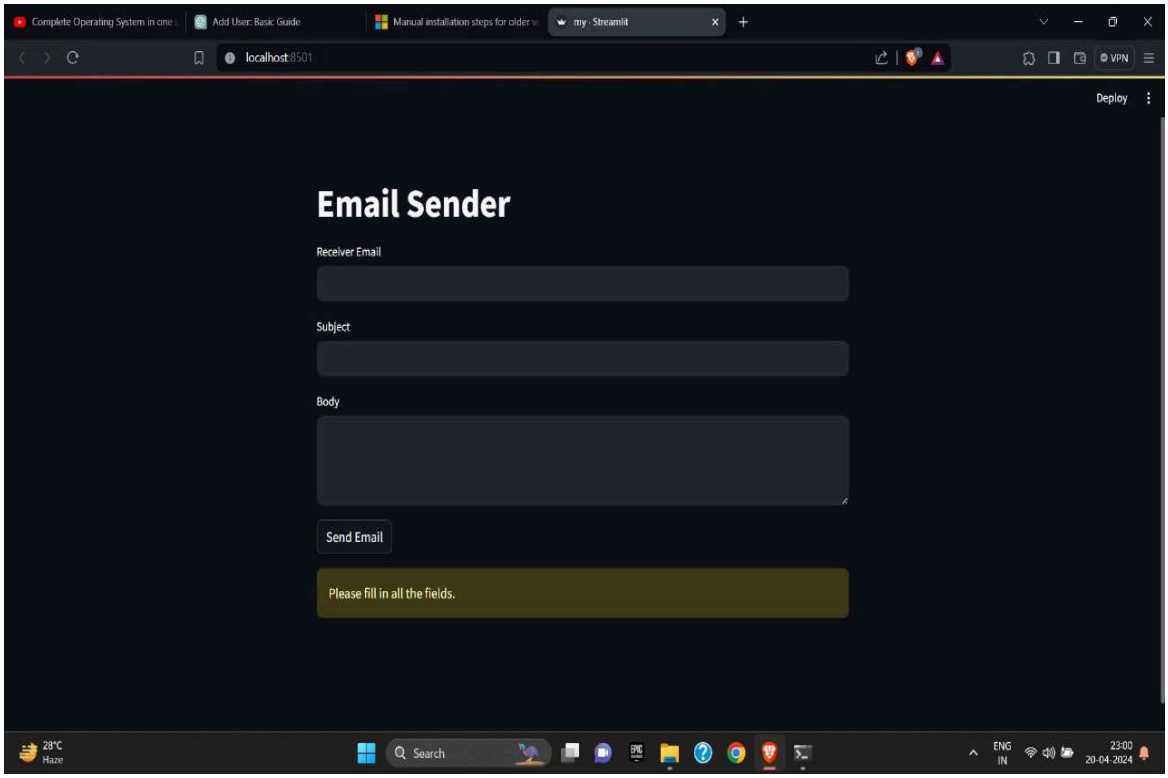
- **Software Used:** Python, Streamlit, MySQL.
- **Hardware Requirements:** Standard computer system with Python environment installed.
- **Programming Languages:** Python for application development, SQL for database operations.

Code :

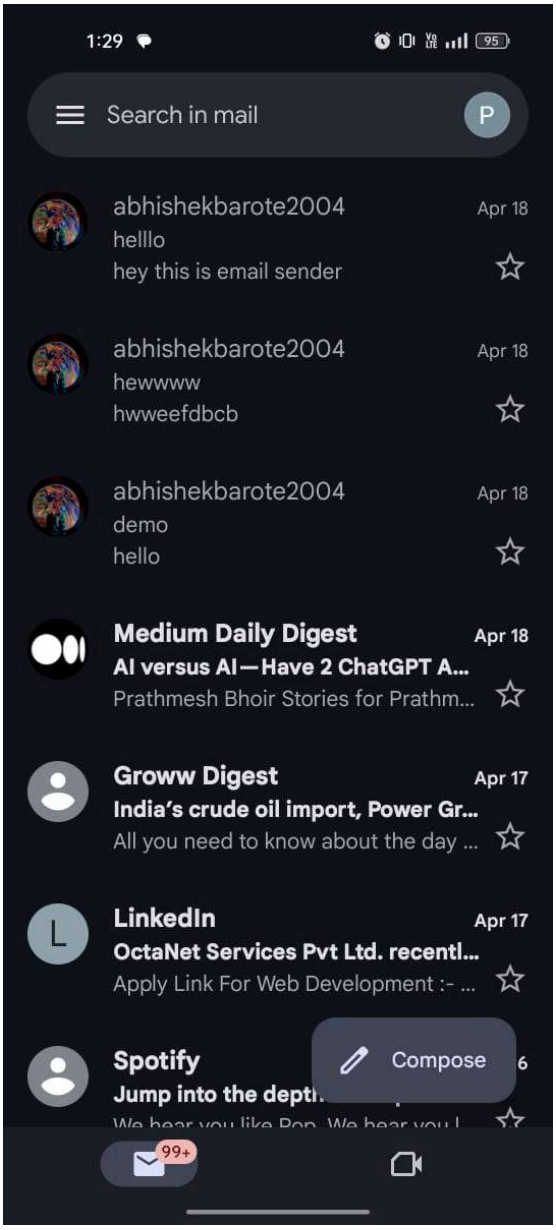
```
import ssl
import smtplib
import streamlit as st
from email.message import EmailMessage

# Hardcoded sender's email and password
SENDER_EMAIL = "xyz@gmail.com"
SENDER_PASSWORD = "pass_word"

def send_email(receiver_email, subject, body):
    em = EmailMessage()
    em['Subject'] = subject
    em['From'] = SENDER_EMAIL
    em['To'] = receiver_email
    em.set_content(body)

    context = ssl.create_default_context()

    with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp:
        smtp.login(SENDER_EMAIL, SENDER_PASSWORD)
        smtp.sendmail(SENDER_EMAIL, receiver_email, em.as_string())

def main():
    st.title("Email Sender")

    receiver_email = st.text_input("Receiver Email")
    subject = st.text_input("Subject")
    body = st.text_area("Body")
```

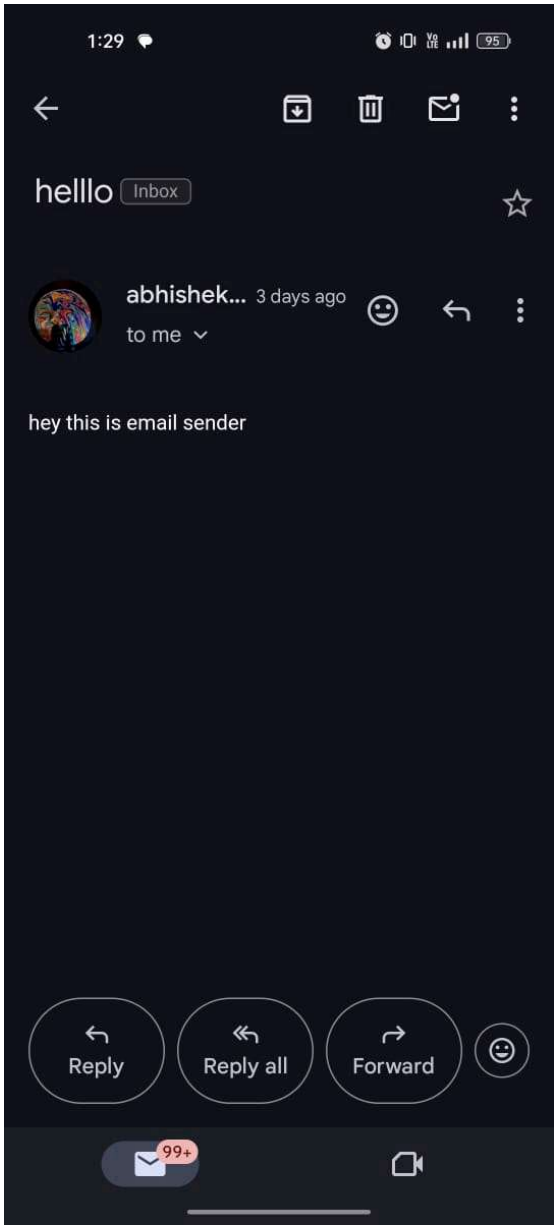
```
    if st.button("Send Email"):
        if receiver_email and subject and body:
            try:
                send_email(receiver_email, subject, body)
                st.success("Email sent successfully!")
            except Exception as e:
                st.error(f"Error occurred: {str(e)}")
        else:
            st.warning("Please fill in all the fields.")

if _name_ == "_main_":
    main()
```

Output :



Interface of Email Sender





After Email Send Successfully..

Results and Conclusion :

The Email Sender project successfully accomplishes its objectives by providing a functional application for composing, sending, and managing emails. The use of Tkinter for frontend development ensures a user-friendly interface, while Django and MySQL facilitate efficient backend operations and data management. Through this project, users can streamline their email sending process and improve productivity. In conclusion, the project demonstrates the effective integration of frontend, backend, and database technologies to create a robust email sending application in Python.

References :

[1] R. Sureswaran, H. A. Bazar, O. Abouabdalla, A. M. Manasrah and H. El-Taj, "Active e-mail system SMTP protocol monitoring algorithm," *2009 2nd IEEE International Conference on Broadband Network & Multimedia Technology*, Beijing, China, 2009, pp. 257-260, doi: 10.1109/ICBNMT.2009.5348490.

[2] D. S, H. T, V. E, R. D. S, S. D. M and P. Sivakumar, "Implementing BeagleBone Black as a Single Board Computer by Transferring E-mail using SMTP," *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, Pudukkottai, India, 2022, pp. 1184-1187, doi: 10.1109/ICACRS55517.2022.10029224.

[3] L. Xie, Yanhua Liu and G. Chen, "A forensic analysis solution of the email network based on email contents," *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Zhangjiajie, China, 2015, pp. 1613-1619, doi: 10.1109/FSKD.2015.7382186.