Automatic Certificate Generation and Emailing System

[#1] Shreya Nahta
Department of Electronics and Communication Engineering,
Nirma University, Ahmedabad
21bec117@nirmauni.ac.in

[#2] Hitesh Gehlot
Department of Electronics and Communication Engineering,
Nirma University, Ahmedabad
21bec047@nirmauni.ac.in

***Abstract*— The proposed system aims to automate the entire procedure, to simplify the certificate creation and distribution process. It speeds up certificate delivery via email, guarantees accuracy, and minimises manual labour by utilising software automation. This creative solution tackles certificate management's issues of convenience and efficiency.**

***Key Words*— Software Automation, Certificate Generation, Emailing System, Convenience, Certificate Management.**

I. Introduction

An Automatic Certificate Generation and Emailing System is a simplified piece of software that makes it easy to make and send certificates to people who finish events or classes. It automatically generates certificates based on models that can be changed, saving time and reducing the chance of making mistakes. Email feature is built into the system so that certificates can be sent to recipients easily and quickly. With user registration, data is kept safe and users are managed. Administrators are kept up to date with notifications and reports, which improves total efficiency. This method makes it easier to give out certificates and can be customized to match the brand of the organization. It is a great tool for schools, training groups, and people in charge of events.

II. Key Libraries Used

1.cv2 (OpenCV):

- Purpose: OpenCV (Open Source Computer Vision) is a library designed for computer vision and image processing tasks.
- Usage in Code: In this script, OpenCV is used for reading, modifying, and saving images. It provides functions for tasks like reading an image file (`cv2.imread()`), adding text to an image (`cv2.putText()`), and saving an image (`cv2.imwrite()`).

2. pandas:

- Purpose: Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like dataframes, which are particularly useful for working with tabular data.
- Usage in Code: The script uses Pandas to read data from an Excel file into a dataframe (`pandas.read_excel()`). Dataframes make it easy to work with structured data, like the student information in the Excel file.

3. smtplib:

- Purpose: The `smtplib` library is part of Python's standard library and is used for sending emails using the Simple Mail Transfer Protocol (SMTP).
- Usage in Code: The script utilizes `smtplib` to set up an SMTP server connection, login, and send emails to recipients with certificates attached.

4. os:

- Purpose: The `os` library provides a way to interact with the operating system, allowing for operations such as file and directory manipulation.
- Usage in Code: In this script, `os` is used to remove temporary files (`os.remove()`) after sending emails, helping to clean up the workspace and save storage.

5.email.mime.multipart,email.mime.text,email.mime.image :

- Purpose: These modules are part of Python's email handling libraries (`email` and `email.mime`). They allow for the creation and manipulation of MIME (Multipurpose Internet Mail Extensions) objects, which are used to structure and format email messages.
- Usage in Code:The script utilizes these modules to construct a multipart email message (`MIMEMultipart()`) with plain text (`MIMEText()`) and image attachments (`MIMEImage()`).

III. Pseudo Code

1. Import Libraries:Import the necessary libraries for working with images, dataframes, email, and file operations.

2. Read Data: Read student data from an Excel file ('name.xlsx') into a Pandas dataframe (`df`).

3. SMTP Configuration: Set up SMTP server configuration including server address, port, username, and password.

4. Convert Data to Lists:Convert specific columns from the dataframe to Python lists (`list_names`, `list_names1`, and `list_emails`).

5. Loop through Students: Iterate through each student's information using a for loop and enumerate to get both the index and values.

6. Image Processing: Read a certificate image, add student names to it using OpenCV's `cv2.putText()`, and save the modified image.

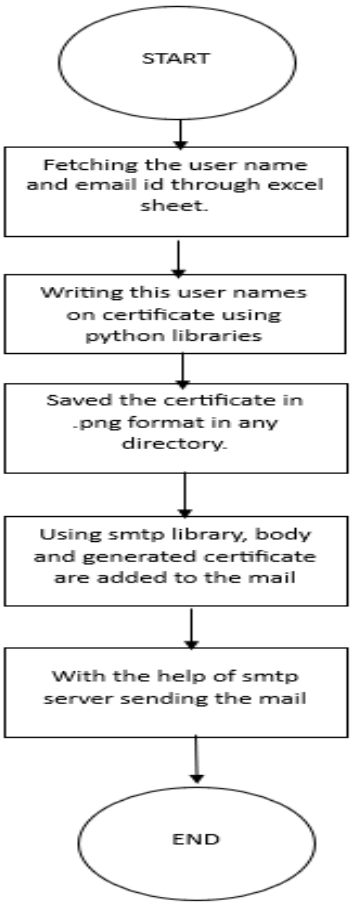
7. *Temporary Certificate File*:Save the modified certificate image as a temporary file.

8. *Send Email:* Use SMTP to send an email with the certificate attachment to the student.

9. *Print Messages*:Print success messages or error messages if an exception occurs during email sending.

10. *Cleanup*: Delete the temporary certificate file after sending the email.

IV. Flow Chart



V. Commands Used In Linux To Import Libraries

1.) sudo apt-get install python3-pip
2.) pip3 install opencv-python
3.) pip3 install pandas
4.) In Linux, the *`os`*, *`smtplib`* libraries are part of the Python standard library, so we don't need to install them separately.

VI. Code With Explanation

```
# Import necessary libraries

import cv2  # OpenCV library for image processing
import pandas  # Pandas library for working with dataframes
import smtplib  # SMTP library for sending emails
import os  # Operating system library for file operations
from email.mime.multipart import MIMEMultipart  # MIME library for email attachments
from email.mime.text import MIMEText  # MIME library for email text
from email.mime.image import MIMEImage  # MIME library for email images

# Read the student data from the Excel file
df = pandas.read_excel('name.xlsx')

# Convert a specific column into a Python list
list_names = df['Name'].tolist()  # Convert the 'Name' column of the dataframe to a Python list
list_names1 = df['Name1'].tolist()  # Convert the 'Name1' column of the dataframe to a Python list
list_emails = df['Email'].tolist()  # Convert the 'Email' column of the dataframe to a Python list

# SMTP email configuration
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587
SMTP_USERNAME = 'hiteshgehlot803@gmail.com'
SMTP_PASSWORD = 'atkb zcme swlh nddb'

# Loop through each student's information
for index, (name1, name2, Email) in enumerate(zip(list_names, list_names1, list_emails)):

# Read the certificate image
  image = cv2.imread('certificate.png')
```

```
    # Add student names to the certificate image
cv2.putText(image, name1, (650, 780), cv2.FONT_HERSHEY_COMPLEX, 3, (0, 0, 0), 1, cv2.LINE_AA)
cv2.putText(image, name2, (730, 1170), cv2.FONT_HERSHEY_COMPLEX, 2, (0, 0, 0), 1, cv2.LINE_AA)

 # Save the modified certificate image with the student's name
 cv2.imwrite(f'D:\sl certificates\{name1}.png', image)

 # Save the certificate as a temporary file
 certificate_path = f'temp_{name1}.png'
 cv2.imwrite(certificate_path, image)

 # Send the certificate via email
 try:
     # Connect to the SMTP server
     server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
     server.starttls()
     server.login(SMTP_USERNAME, SMTP_PASSWORD)

     # Create an email message
     msg = MIMEMultipart()
     msg['From'] = SMTP_USERNAME
     msg['To'] = Email
     msg['Subject'] = 'Certificate of Completion'

     # Add a plain text body to the email
     body = f'Dear {name1},\n\nCongratulations on
     completing your course! Please find your certificate
      attached.'
     msg.attach(MIMEText(body, 'plain'))

     # Attach the certificate image to the email
     with open(certificate_path, 'rb') as img_file:
     image = MIMEImage(img_file.read(),
     name=f'{name1}.png')
     msg.attach(image)

     # Send the email
     server.sendmail(SMTP_USERNAME, Email,
     msg.as_string())
     server.quit()

     # Print a success message
     print(f'Processing Certificate{index+1}/{len(list_names)}
    - Sent to {Email}')
     except Exception as e:
 # Print an error message if sending fails
  print(f'Error sending certificate to {Email}: {str(e)}')

 # Delete the temporary certificate file
 os.remove(certificate_path)
```

VII. INPUT

Fig. 1. Sample Certificate

Name	Name1	Email
Shreya Nahta	Vaishali Dhare	21bec117@nirmauni.ac.in
Vinit Arora	Amisha P Naik	21bec136@nirmauni.ac.in
Avadhi Jain	Yogesh Trivedi	21bch007@nirmauni.ac.in
Hitesh Gehlot	Vaishali Dhare	21bec047@nirmauni.ac.in
Jay Vadodariya	Amisha P Naik	21bec130@nirmauni.ac.in
Yashvi Singhal	Vaishali Dhare	21bec140@nirmauni.ac.in
Giriraj Rathi	Vaishali Dhare	21bec034@nirmauni.ac.in
Dhruv Shah	Yogesh Trivedi	21bec030@nirmauni.ac.in
Mittal Kalal	Amisha P Naik	21bec067@nirmauni.ac.in
Shreedhar Joshi	Vaishali Dhare	21bee039@nirmauni.ac.in

Fig. 2. Excel Sheet

VIII. Output

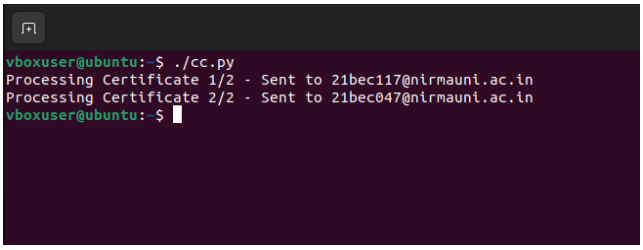


Fig.3 Output

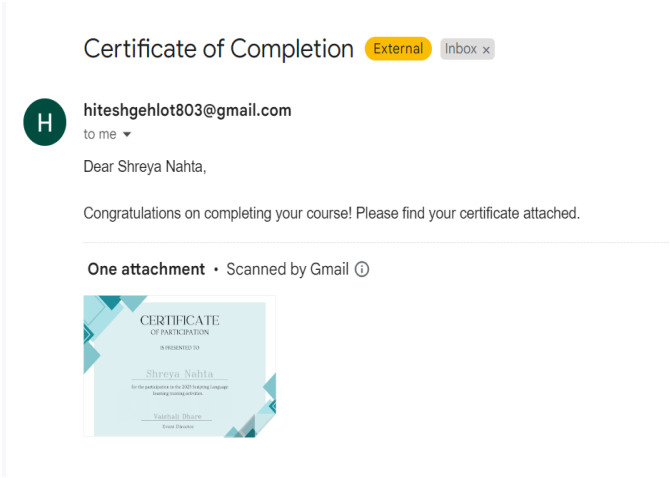


Fig. 4. Confirmation Mail

Fig. 5. Generated Certificate

IX. Conclusion

In conclusion, the implementation of an Automatic Certificate Generation and Emailing System represents a significant leap forward in efficiency and convenience. By seamlessly automating the once time-consuming process of certificate creation and distribution, this system not only streamlines administrative tasks but also ensures accuracy and timeliness, reducing the likelihood of errors and delays.It a valuable asset for any institution or business, saving time, reducing costs, and enhancing overall operational efficiency.

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 Certificate Generation and Emailing System.

.

References

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

[2] M. F. Sarker, and S. Washington, Learning Python Network Programming. Packt Publishing Ltd., 2015.

[3] M. Driscoll, Pillow: Image Processing with Python. Independently Published, 2021

[4] J. Goerzen, “Simple message transport protocol,” in Foundations of Python Network Programming. Berkeley, CA: Apress, 2004. [Online]. Available: https://doi.org/10.1007/978-1-4302-0752-8_10

[5] J. Viega, M. Messier, and P. Chandra, Network Security with OpenSSL: Cryptography for Secure Communications. O’Reilly Media, Inc., 2002.