Development of a software platform for e-certificate generation and auto-sending

[1]Aditya Agrawal, [2]Parth Parikh, [3]Dhruvi Patel, [4]Vaishali Dhare
[1,2,3,4] Electronics and Communication Engineering Department, Institute of Technology, Nirma University, Ahmedabad, India
[1]20bec003@nirmauni.ac.in, [2]20bec081@nirmauni.ac.in, [3]20bec026@nirmauni.ac.in, [4]vaishali.dhare@nirmauni.ac.in

***Abstract*—Nowadays automation of every task is needed to save time and ease of completing the task. In this context, the software platform is developed to generate the certificates. In this paper, a single platform using Python is developed which takes care of generating certificates in bulk and sending them to the concerned person. This is useful when the number of participants in the workshop, seminar, event, etc. is more. The developed algorithm is simple yet effective.**

Keywords: e-certificate, software platform, Python

I. INTRODUCTION

The prime objective of the campaign launched by the Indian Government a few years ago called "Digital India" is to make the country digitally empowered in the field of technology. This paper is in line with this statement by making the automation of e-certificate generation. In this paper, a single software platform is developed which generates not only an e-certificate from a database but also sends the generated e-certificate to the concerned person. This platform is useful for conferences, workshops, events, etc. where the number of participants is more, and the process of making and sending a certificate manually is time-consuming.

Google [1] has this facility but requires more than one software like google forms, Microsoft Look, etc. which is time-consuming. In this paper, only one platform is required which takes care of everything.

In [2], the certificate generation system is proposed. In [2], only the certificate generation is possible but sending it to the concerned person is not explored in it. Microsoft Word [3] is providing the facility for auto-generation and auto-sending facility but it is paid platform. We aim to develop an open-source platform so that all can easily use it. Many open-source platforms are available but some of them required multiple software to complete the whole process. Table 1 shows the available certificate generation and auto-emailing software/platforms [4-6].

Table 1. Certificate generation and auto emailing platform list

Sr. No	Certificate Generation	Auto-Emailing
1.	Canva	Drip
2.	Certificate Magic	Keap
3.	Certified	Google Sheets
4.	Smart Certificate	Sendinblue
5.	Microsoft Word	ActiveCampaign
6.	Certify'em	Omnisend
7.	Mettl Certify	AWeber
8.	SmartDraw	Emma

In this paper, the platform is developed using Python. This single platform is generating and auto-emailing the certificates. This platform is not only restricted to certificate generation but also applicable to the marksheet, grade reports, transcripts, etc. generation and auto emailing.

The paper is organized as follows: Section II describes the used Python libraries. Platform development is described in Section III. The results are discussed in Section IV. The paper concludes in Section V.

II. BACKGROUND

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics [7-9]. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components. Normally, a library is a collection of books or a room or place where many books are stored to be used later. Similarly, in the

programming world, a library is a collection of precompiled codes that can be used later on in a program for some specific well-defined operations. Other than pre-compiled codes, a library may contain documentation, configuration data, message templates, classes, values, etc. A Python library is a collection of related modules. It contains a bundle of codes that can be used repeatedly in different programs. It makes Python Programming simpler and more convenient for the programmer. As we don't need to write the same code again and again for different programs. Python libraries play a very vital role in the fields of Machine Learning, Data Science, Data Visualization, etc. To solve the above-stated problem statement in the abstract, we have made a Python program using some Python libraries like O**penCV** [10] via cv2 which is used for computer vision, machine learning, and image processing and now plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. O**penpyxl** [11] library is used to read from an Excel file or write to an Excel file. Data scientists use Openpyxl for data analysis, data copying, data mining, drawing charts, styling sheets, adding formulas, and more. we can generate multiple certificates using an Excel sheet.

III. PLATFORM DEVELOPMENT AND IMPLEMENTATION

The pseudo-code for certificate generation is shown in Algorithm 1.

Algorithm-1 Certificate Generator

1. Import (cv2, openpyxl) for real-time operation and read/write into an Excel file

2. Provide a details path to grab the name and other details of the Excel sheet & an output path to store the generated certificates

3. **for** i = 2 to 11

    grab rows = i and corresponding columns

    C1- containing the names of winning students

    C2- winning position of student

    C3- event name in which student won

    C4- university name of winning student

5. Assign template path to image read function via *cv.imread* function

6. Assign font style to each column in Excel sheet

7. Put each column of the Excel sheet into the certificate template via the *cv.putText* function.

8. Save the certificates generated to a specified path which is to be mentioned in *the cv.imwrite* function

10. **end**

First of all, libraries like **OpenCV** via cv2, and **openpyxl** is used for real-time operation and to read from an Excel file or write into an Excel file respectively. Further, the template path is passed – to get access to the certificate template, the details path- to grab the name and other details from the Excel sheet, and the output path- to store the generated certificates, are provided in the code. To get the above-stated details in the desired size and color we set the size and color in (B, G, R) format. Then the Excel file is loaded to **the obj** variable through load_workbook() which is an inbuilt function of the openpyxl library. This Excel file is activated through obj.active to do further manipulation. To print the 'x' number of certificates, we use for loop statement as, for i in range (2, x+1):

Inside the for loop, to get the name of the candidate who won to be printed on the certificate we grab the 'i' number of rows and number of columns (=1) from the Excel file, and we use the sheet.cell and get_name.value() function as mentioned in the code.

To get the winning position of the candidate, we grab the 'i' number of rows and the number of columns (=2) from the Excel file, and we use a sheet.cell and get_position.value() function.

To get the event name in which the candidate won, we grab the 'i' number of rows and number of columns (=3) from the Excel file, we use the sheet.cell and get_event.value() function.

To get the college name of the candidate to be printed on the certificate, we grab the 'i' number of rows and number of columns (=4) from the Excel file, we use a sheet.cell and get_clg.value() function.

For extracting the certificate template, we use cv.imread() (inside which template path is to be provided)which is an inbuilt function of the cv2 library. Font style is chosen via cv. Font() function. To get the size of the name to be printed, we need to know the (x,y) pixels where the name is to be written on the template, for which the cv.putText () function is used.

At last to save the certificates generated, cv.imwrite is used in which (f ' path of folder/ certificate name variable,img) is to be given.

The pseudo-code for auto-emailing is shown in Algorithm 2.

Algorithm-2 Email generator and sender

1. Import libraries (smtplib, os, email.mime) to send emails, and to send attachments along with text in emails.

2. Mention the email id of the sender and list of receivers

3. Open mail-> security & privacy -> app passwords -> custom search SMTP Email-> 16-digit alpha password

4. Assign a subject and auto-generated password to variable

5. Provide the path of the folder containing file attachments

6. function send_mails (email_list)

7. **for** the filename in the file list

8. **for** names in email_list

9. Enter the text which is to be sent with the attachments using MIMEMultipart inbuilt function to define parts of the email

10. Open the file attachment in read binary format and to send attachments use encoding base 64.
11. Made downloadable attachments for users, we load different bits via the set_payload function & later in ASCII characters encode_base64.
12. To fire the email successfully connect with the server turn an existing insecure connection to a secured one via the starttls function and then print (sending email to the {person})
13. next
14. end
15. call the function send_mails (email_list)

To send an email firstly we need to import certain libraries i.e. Simple Mail Transfer Protocol (SMTP), and OS library to read the name of the file which are to be sent from a particular path. Next, we import the MIME library. MIME is an email package used for managing email messages. The overall structure of the email package can be divided into three major components.

1. Subject of the email,
2. Body of the email,
3. Attachment to be sent in email.

We also import the email portion of the encoders library because to fire an email we need to encode the package in a 64-Encoder.

SMTP library has various features. To use the email feature we take port 587, which is a standard secure SMTP port.

We have created a function send_mail(). So that it can be used repeatedly without writing the whole code again & again.

We define a list email_list which contains a list of people to whom the email has to be sent from the dummy email that we have created "sl2ecde55@gmail.com". Then we define variables to store the Body of the email, the Subject of the email and a

FILEList variable to access all the attachments.

A for loop is created to send the email to each person in the list defined as "email_list". Then we create a msg object which will have a different part of the email i.e. email_from, email_list, subject using MIME object. We attach the body of the email to this using msg. attach (MIMEText) function. An attachment variable defines that all the attachments from the given path are read in binary form. Furthermore, to let the receiver able to download the attachment we create an attachment package and load different bits so that it is downloadable. The encoded file is in ASCII characters format. So, we convert it back to string and store it in variable text. Now the email is ready to get fired!!

To connect with the SMTP server, we need to give the login credentials i.e. smtp_port, smtp_server, email_from, and app password for SMTP Email, and lastly start the TLS for enhanced encryption in transit. After successfully sending the email we quit the server.

IV. Results and Discussion

The Python code is developed and runs using Jupiter Notebook IDE. The sample certificate generated by the platform is shown in Fig. 1. Fixed template of the certificate is to be provided to the code. It automatically filled up the data like name, affiliation, grade, etc. from the database. Any number of fields can be filled up using the developed code.

Fig. 1: Sample certificate generated using the details in the excel sheet.

The individual certificate is generated as an image or pdf. Further, each generated certificate will be sent to the concerned as an attachment with a fixed email subject and body. The email id will be fetched from the database. Fig. 2 shows the auto generate d email and attachment.

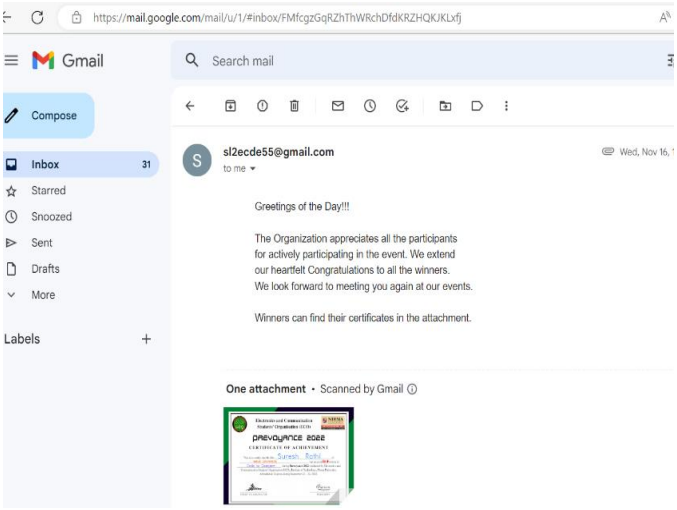

Fig. 2: Automatic email with the certificate as an attachment.

V. Conclusion

In this paper, the software platform is developed using Python. This platform takes care of generating certificates automatically from the database. It was also helpful to send the generated certificate to the concerned participants. Only a single platform is required to perform the certificate generation and sending task. It can be made as an open source with the GUI.

References

[1] https://digitalinspiration.com/docs/document-studio/google-forms/

[2] Shimpi, Srushti A., Sanket Mandare, Aman Trivedi, and Tyagraj Sonawane. "Certificate generation system." International Journal on Recent and Innovation Trends in Computing and Communication, vol. 2, no. , (2014), pp. 380-383.

[3] https://techcommunity.microsoft.com/

[4] https://certifier.me/blog/the-best-online-software-to-create-certificates-9-professional-certificate-makers

[5] https://medium.com/analytics-vidhya/generate-certificates-using-python-a7685985ed77

[6] https://www.courier.com/blog/three-ways-to-send-emails-using-python-with-code-tutorials/

[7] Ganesh SanjivNaik, "Learning Linux Shell Scripting," Packet Publishing Ltd.

[8] Guido Van Rossum and Fred L. Drake Jr, "An Introduction to Python," Network Theory Ltd.

[9] Hans Petter Langtangen, "A Primer on Scientific Programming with Python," Springer

[10] https://docs.opencv.org/4.x/d5/de5/tutorial_py_setup_in_windows.html

[11] https://medium.com/women-who-code-data-science/python-script-to-generate-certificates-6fc0bfe4f2d7