

CMPSC 487W: Software Engineering Design (Fall 2022)

**Project**

**Password Protected Certificate Generator**

Harsh Patel, Andy Huang, and Reuel Manukonda

Alejandro Trofimoff

Submitted On: 5 December 2022

Table of Contents

[Introduction 3](#_Toc121134402)

[Requirement Analysis 3](#_Toc121134403)

[Data Dictionary 4](#_Toc121134404)

[Data Structures 4](#_Toc121134405)

[Design 5](#_Toc121134406)

[Functionalities 6](#_Toc121134407)

[Password-Protected Certificate 6](#_Toc121134408)

[Email 6](#_Toc121134409)

[Algorithm 6](#_Toc121134410)

[Results 7](#_Toc121134411)

[Software Quality Metrics 9](#_Toc121134412)

[Productivity 9](#_Toc121134413)

[Portability 9](#_Toc121134414)

[Iterative Development Method 9](#_Toc121134415)

[Software Testing Strategies 10](#_Toc121134416)

[Conclusion 11](#_Toc121134417)

[User Manual 11](#_Toc121134418)

[Demo Video: 11](#_Toc121134419)

[Code 14](#_Toc121134420)

[References 21](#_Toc121134421)

# Introduction

The project idea for a password protected certificate generator was given by Christine Cicio, the Director of Continuing Education at Penn State Abington. Her current problem was that the department issues password-protected digital certificates to students that complete specific course and that they have to manually create each certificate individually as well as email them to the student. Christine wanted a more efficient solution to create a mass number of certificates in a short period of time.

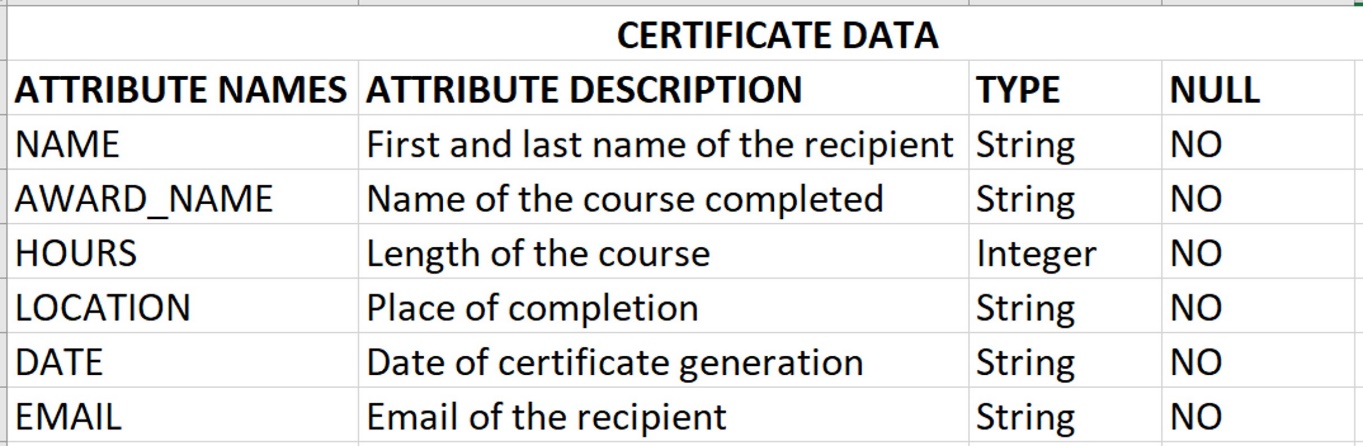
GitHub link: <https://github.com/harsh012000/Password-Protected-Certificate-Emailer>

# Requirement Analysis

* Customizable password protected (read only, no edit) Certificate.
* Automatically emails the certificate to appropriate students.
* Certificate Features:
  + Penn State Branding
  + Student Name
  + Course Completed
  + Date of Completion
  + Length of the Course (Hours)
* Signature of Issuer
* Take Certificate Attributes from user and generate it based on those attributes.
* Example Certificate:



# Data Dictionary



# Data Structures

1. Excel Spreadsheet: User-provided Data for certificates, it contained all the information required to generate a certificate.
2. Data Frame data structure was used to get the data from the excel spreadsheet. Data Frame is a Pandas (Python Library) two-dimensional data structure with columns, much like a table. It was used to gather the fields that needed to be mapped into a certificate (a word document).
3. A dictionary was used to map the data from the entry form to the certificate template as there was only one set of data at a time. A dictionary is much more efficient than a data frame in this case.

# Design

We have two main functions; generate\_from\_excel and open\_window. The generate\_from\_excel function ask the user to select an excel file which it then reads to create a for loop that checks each row and column of a specific name. If the format of the excel file’s format is correct, we find the root directory of a word document template where each data within a row is appended to its corresponding position to create a new word document which is then turned into a PDF and then again, another new PDF that is encrypted. The open\_window function is a Graphical User Interface that has set of varying inputs that match what is needed within the certificate. A while loop is ran to keep the interface open where upon clicking generate certificate an if statement is ran to extract all of the data from the inputs and puts them into variables that are used to call the root directory of the word document template where those variables are mapped to a corresponding position. This new word document is then turned into a PDF and then another new encrypted PDF

# Functionalities

## Password-Protected Certificate

The Encrypt function encrypts a pdf using a user password which is used for opening the document. The function is also responsible for managing user permissions. It blocks the recipient from editing the certificate using tools like Adobe Acrobat but allows the recipient to print it. This adds a layer of security which restricts a recipient from generating different certificates that they haven’t earned.

## Email

When using an excel file, email feature is automatically enabled as there may be 40 or more entries in one file. If manual entry is selected, then there is an option to send an email. The Email feature takes the encrypted pdf, attaches it to an email, adds body text and in a few seconds the recipient receives an email.

# Algorithm

1. Take input from user: Excel File or Entry Form
2. Map each attribute from the input to the Certificate template and save it as word document.
3. Convert the word doc to a PDF and PDF to an Encrypted PDF.
4. Email it to each recipient if there is more than one.

Structure of the Algorithm

There are three main functions used in the program. First, encrypt which is responsible for encrypting a PDF. Second, input from Excel which is responsible for opening the Excel file, mapping the fields from excel file to template, encryption is done here for individual PDFs and emailing. Third, input from Entry Form which is responsible for takin input from the client, mapping the fields from form to template, encryption is done here for a PDF and an email is sent.

Time Complexity

Certificate Generation and Email using Excel File: O(n).

* Depends on the number of records in the excel file.

Certificate Generation and Email using Entry Form: O (1).

* Constant time because single Certificate is generated using Dictionary.

# Results

The program generates an encrypted certificate as a PDF file and emails it to the recipient with the password to open it.

The following is an example output for John Doe (This is what the recipient will see in their outlook).

Text, letter

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

# Software Quality Metrics

## Productivity

Overall, the process is very fast from start to finish compared to manually generating and emailing a large number of certificates.

|  |  |
| --- | --- |
| Task | Time |
| Selecting Excel file/Entering Details | Depends on user speed of typing/Locating file. |
| Processing (Generating Word and Pdf files) | Max 5 minutes if there are lot of entries in an excel file. |
| Emailing | Takes few seconds (Long if first time) |

## Portability

The program effectively works on a Windows computer as long as there is internet connection. Tools like Wine bottler can be used for this program to work on a Mac Operating System.

# Iterative Development Method

The Software development method used for this project was Iterative method. There were two major iterations done during the course of the project:

* October 26, 2022: This involved demonstrating the GUI with the ability to generate a single encrypted certificate when details are entered in an entry form.
* November 11, 2022: This involved demonstration of the entire program with getting data from excel and sending multiple emails.

# Software Testing Strategies

The testing strategies used were:

* Unit Testing: To test individual components of the Software, like certificate creation from the entry form and input from excel file, Encrypting PDF, and Email.
* Integration Testing: To check how well entry form/Excel input works with encryption and Email functionalities.
* System Testing. It is the process to evaluate how the various components of an application interact together in the full, integrated system or application. First, the GUI with only entry form was tested to see how well it works using several different inputs and checking if the program fails. Many errors were encountered including file not found and invalid file paths, all was fixed. After the success result, the GUI with the Excel file as an input was tested, that also had few errors similar to the prior testing, all were fixed and now both efficiently produce encrypted certificates. The last aspect was email. After its implementation, the program was put to the test several times to check if the email appears to be exactly what the client wants it. Some body text was changed, and certificate naming was changes to achieve conclude the testing.
* Acceptance Testing: To test how well the Software meets the client requirement. Client Satisfaction with the Software was the key to acceptance testing.

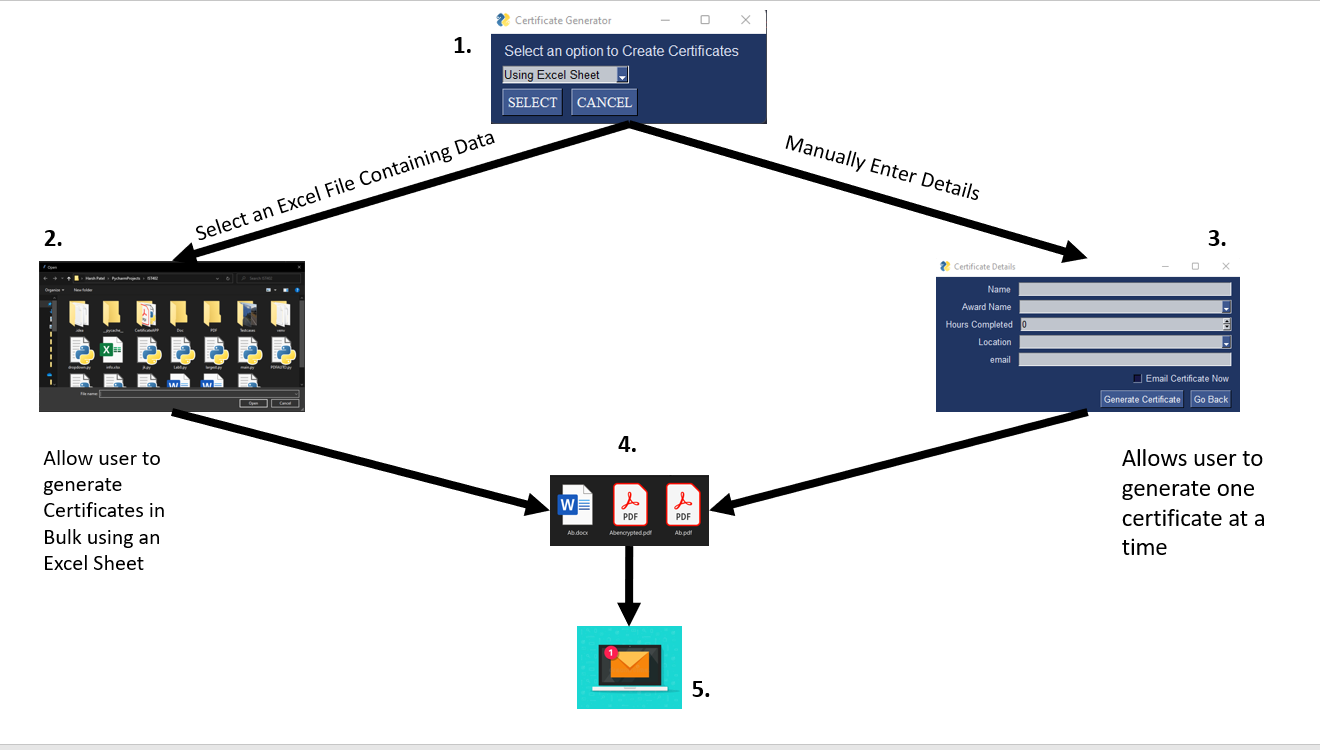
# Conclusion

* This was project was the first project involving a client. We learned a lot about Project Management and working with clients. Other aspects learned include Efficiently develop a GUI in python, gained more knowledge on handling files and file paths, learned to automate a manual process by using a single programming language, and How to send an email with an attachment with Python. Some improvements that can be made to make this a better program include: reduce memory usage, 2-Factor Authentication, making it cross-platform (MAC) as currently it only works on Windows.

# User Manual

## Demo Video:

[](https://www.youtube.com/embed/pRpyZMF-c-I?feature=oembed)



When the application is opened, there is a drop-down menu which allows the user the pick how to proceed.

Default option: From an Excel sheet, upon selection the user can select a file (Excel Spreadsheet) which contains the data for the students who completed a course.

After the Excel file is selected from the popup file explorer the program takes control.

Option 2: Manual Entry, upon selection the user will see an entry form where the user can enter the details of a particular recipient.

After the Generate Certificate button is pressed, the programs do the rest.

The program reads the input Excel file or the entered values, then a word document is generated, followed by a PDF, and finally an encrypted PDF. The program deletes the PDF copy and only keeps the Word document and the Encrypted PDF. The program then emails the Encrypted PDF to each recipient if there are more than one entry. The email’s body is generated using html with required links.

Note: The user will have a Word Document and Encrypted PDF versions of the Certificates in the Application folder. This is done in case the user wants to make some changes and re-send it.

Technical Details

User Password: Password to open the PDF. Randomly generated – different for each recipient.

Owner Password: Password to Edit the PDF. PSUabCE2022

Certificate Template: Saved as Tem.docx. The user can change most of the template except the fields that are mapped through the excel file.

Excel File: Must be in a specific format. It should only have the fields that the template can take. That is the text in { } in the template.

Note: After the program has successful finished the process, the certificates can be deleted to save storage. As all the Certificates will be in the Email attachments to view.

# Code

*import* PySimpleGUI *as* sg

*from* PySimpleGUI *import* filedialog

*from* pathlib *import* Path

*import* datetime

*from* docxtpl *import* DocxTemplate

*from* docx2pdf *import* convert

*from* win32com *import* client

*import* pandas *as* pd

*import* os

*import* time

*from* hashlib *import* md5

*from* PyPDF4 *import* PdfFileReader, PdfFileWriter

*from* PyPDF4.generic *import* NameObject, DictionaryObject, ArrayObject, \

    NumberObject, ByteStringObject

*from* PyPDF4.pdf *import* \_alg33, \_alg34, \_alg35

*from* PyPDF4.utils *import* b\_

*from* email.message *import* EmailMessage

*import* smtplib

*from* email.mime.multipart *import* MIMEMultipart

*from* email.mime.text *import* MIMEText

*from* email.mime.base *import* MIMEBase

*from* email *import* encoders

*from* password\_generator *import* PasswordGenerator

*# Pdf Encryption*

def encrypt(writer\_obj: PdfFileWriter, user\_pwd, owner\_pwd=None, use\_128bit=True):

*"""*

*Encrypt this PDF file with the PDF Standard encryption handler.*

*:param str user\_pwd: The "user password", which allows for opening*

*and reading the PDF file with the restrictions provided.*

*:param str owner\_pwd: The "owner password", which allows for*

*opening the PDF files without any restrictions.  By default,*

*the owner password is the same as the user password.*

*:param bool use\_128bit: flag as to whether to use 128bit*

*encryption.  When false, 40bit encryption will be used.  By default,*

*this flag is on.*

*"""*

*import* time, random

*if* owner\_pwd == None:

        owner\_pwd = user\_pwd

*if* use\_128bit:

        V = 2

        rev = 3

        keylen = int(128 / 8)

*else*:

        V = 1

        rev = 2

        keylen = int(40 / 8)

*# permit copy and printing only:*

    P = -44

    O = ByteStringObject(\_alg33(owner\_pwd, user\_pwd, rev, keylen))

    ID\_1 = ByteStringObject(md5(b\_(repr(time.time()))).digest())

    ID\_2 = ByteStringObject(md5(b\_(repr(random.random()))).digest())

    writer\_obj.\_ID = ArrayObject((ID\_1, ID\_2))

*if* rev == 2:

        U, key = \_alg34(user\_pwd, O, P, ID\_1)

*else*:

*assert* rev == 3

        U, key = \_alg35(user\_pwd, rev, keylen, O, P, ID\_1, False)

    encrypt = DictionaryObject()

    encrypt[NameObject("/Filter")] = NameObject("/Standard")

    encrypt[NameObject("/V")] = NumberObject(V)

*if* V == 2:

        encrypt[NameObject("/Length")] = NumberObject(keylen \* 8)

    encrypt[NameObject("/R")] = NumberObject(rev)

    encrypt[NameObject("/O")] = ByteStringObject(O)

    encrypt[NameObject("/U")] = ByteStringObject(U)

    encrypt[NameObject("/P")] = NumberObject(P)

    writer\_obj.\_encrypt = writer\_obj.\_addObject(encrypt)

    writer\_obj.\_encrypt\_key = key

document\_path = Path(\_\_file\_\_).parent / "Tem.docx"  *# use the template here*

doc = DocxTemplate(document\_path)

date = datetime.datetime.today()

*# set the theme for the screen/window*

sg.theme('DarkBlue13')

def generate\_from\_excel():

*try*:

        file\_path = filedialog.askopenfilename()

        word\_app = client.Dispatch("Word.Application")

        data\_frame = pd.read\_excel(file\_path)

*for* r\_index, row *in* data\_frame.iterrows():

            name = row['Name']

*# print(name)*

            award\_name = row['Award\_Name']

            hours = row['Hours']

            location = row['Location']

            date = row['Date']

            email\_address = row['Email']

            tpl = DocxTemplate("Tem.docx")

            df\_to\_dcot = data\_frame.to\_dict()

            x = data\_frame.to\_dict(orient='records')

            context = x

            tpl.render(context[r\_index])

*'''if os.path.isfile(name + ".docx"):*

*os.rename(name + ".docx", name + r\_index + ".docx")*

*tpl.save( name + r\_index + ".docx")*

*else:*

*tpl.save(name + ".docx")'''*

            r\_index = str(r\_index)

            tpl.save(name + r\_index + ".docx")

            time.sleep(1)

*# get project folder path*

            ROOT\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_))

*# convert docx to pdf*

            doc = word\_app.Documents.Open(ROOT\_DIR  + '//' + name + r\_index + '.docx')

            print('Exporting')

            doc.SaveAs(ROOT\_DIR  + '//' +  name + r\_index + '.pdf', FileFormat=17)

            unmeta = PdfFileReader(ROOT\_DIR  + '//' + name + r\_index + '.pdf')

            writer = PdfFileWriter()

            writer.appendPagesFromReader(unmeta)

            pwo = PasswordGenerator()

*#creates a randomly generated password for the student to acess the encrypted PDF*

            user\_pass = pwo.shuffle\_password('qwertyuiopasdfghjklzxcvbnm1234567890', 5)

*#a set password for the creator of the PDF*

            owner\_pass = 'PSUabCE2022'

            encrypt(writer, user\_pass, owner\_pass)

            encrypted\_pdf = ROOT\_DIR  + '//' + name + r\_index + "encrypted.pdf"

            os.remove(ROOT\_DIR  + '//' +  name +  r\_index + '.pdf')

*with* open(encrypted\_pdf, 'wb') *as* fp:

                writer.write(fp)

*#sg.popup("File saved", f"File has been saved here: {encrypted\_pdf}")*

*# Email Code*

            msg = MIMEMultipart()

            sender = "testtest1205@outlook.com"

            password = "abc1234567890"

            receiver = email\_address

*'''body = f"Congratulations {name} on your {award\_name}!!" \*

*f"\nYour official Penn State Certificate is attached\nYour password for your certificate is:  " + user\_pass'''*

            body =  f'<pre>On behalf of the <a href="https://www.abington.psu.edu/continuing-education">Penn State Abington Continuing Education</a> team,\

congratulations on the completion of the {award\_name} program. \

You will find your certificate of program completion attached to this email. \

The password to unlock your certificate is: <u>{user\_pass}</u>. If you have any questions, please contact us at abce@psu.edu. \

Stay connected with us on <a href="https://www.linkedin.com/company/penn-state-abington-continuing-education/?viewAsMember=true ">LinkedIn</a> \

and <a href = "https://www.facebook.com/PSUAbingtonCE/">Facebook<a/>.<pre>'

            msg.attach(MIMEText(body, 'html'))

            msg['Subject'] = f'{award\_name} Certificate Completion'

            msg['From'] = sender

            msg['To'] = receiver

            binary\_pdf = open(encrypted\_pdf, 'rb')

            payload = MIMEBase('application', 'octate-stream', Name = name + r\_index + '.pdf')

            payload.set\_payload(binary\_pdf.read())

*# enconding the binary into base64*

            encoders.encode\_base64(payload)

*# add header with pdf name*

            payload.add\_header('Content-Decomposition', 'attachment', filename = name + r\_index + '.pdf')

            msg.attach(payload)

*with* smtplib.SMTP('smtp.office365.com', 587) *as* server:

                server.starttls()

                server.login(sender, password)

                server.sendmail(sender, receiver, msg.as\_string())

*#Pop up verifying email was sent*

*#sg.popup("Certificate Has been emailed to: " + email\_address)*

        sg.popup("All Certificates have been emailed")

        word\_app.Quit()

*except* Exception *as* e:

        sg.Popup("Excel format Incorrect")

def open\_window():

    layout = [

        [sg.Text("Name"), sg.Input(size=(45, 20), key="Name", do\_not\_clear=False)],

        [sg.Text("Award Name"),

         sg.Combo(['Digital Marketing Professional','Diversity, Equity, Inclusion, and Belonging in the Workplace','Nursing Care Home Administrator','Personal Care Home Administrator','Project Management','Public Entity Leadership Development','Trauma Informed Practices for Educators'], size=(43, 20), key="Award\_Name")],

        [sg.Text("Hours Completed"),

         sg.Combo(['2','7','15','30','49','84','100','120'], size=(43, 20), key="Hours", )],

        [sg.Text("Location"), sg.Combo(['Virtual', 'Delaware Valley Trusts'], size=(43, 20), key="Location")],

        [sg.Text("email"), sg.Input(key="Email", size=(45, 20), do\_not\_clear=False)],

        [sg.Checkbox("Email Certificate Now", default=False, key="Emailed Certificate")],

        [sg.Button("Generate Certificate"), sg.Exit("Go Back")],

    ]

    window = sg.Window('Certificate Details', layout, modal= True, element\_justification="right", resizable = True)

    choice = None

*while* True:

        event, values = window.read()

*if* event == sg.WIN\_CLOSED or event == "Go Back":

*break*

*if* event == "Generate Certificate":

            print(event, values)

            values["Date"] = date.strftime("%B %Y")

            doc.render(values)

            output\_path = Path(\_\_file\_\_).parent / f"{values['Name']}.docx"

            doc.save(output\_path)

            pdf\_output\_path = Path(\_\_file\_\_).parent / f"{values['Name']}.pdf"

            convert(output\_path, pdf\_output\_path)

            sg.popup("File saved", f"File has been saved here: {pdf\_output\_path}")

            time.sleep(1)

*# get project folder path*

            ROOT\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_))

            unmeta = PdfFileReader(ROOT\_DIR + f"//{values['Name']}.pdf")

            writer = PdfFileWriter()

            writer.appendPagesFromReader(unmeta)

            pwo = PasswordGenerator()

*#creates a randomly generated password for the student to acess the encrypted PDF*

            user\_pass = pwo.shuffle\_password('qwertyuiopasdfghjklzxcvbnm1234567890', 5)

*#a set password for the creator of the PDF*

            owner\_pass = 'PSUabCE2022'

            encrypt(writer, user\_pass, owner\_pass)

            encrypted\_pdf = ROOT\_DIR + f"//{values['Name']}encrypted.pdf"

            os.remove(ROOT\_DIR  + '//' + values['Name'] + '.pdf')

*with* open(encrypted\_pdf, 'wb') *as* fp:

                writer.write(fp)

            sg.popup("File saved", f"File has been saved here: {encrypted\_pdf}")

*# Email Code*

*if* values["Emailed Certificate"] == True:

                    msg = MIMEMultipart()

                    sender = "testtest1205@outlook.com"

                    password = "abc1234567890"

                    receiver = values['Email']

*'''body = f"Congratulations {values['Name']} on your {values['Award\_Name']} with a total of {values['Hours']} hours at {values['Location']}!!" \*

*f"\nYour official certificate is attached" + "\nYour password for your certificate is:  " + user\_pass '''*

                    body = f'<pre>On behalf of the <a href="https://www.abington.psu.edu/continuing-education">Penn State Abington Continuing Education</a> team,\

congratulations on the completion of the {values["Award\_Name"]} program. \

You will find your certificate of program completion attached to this email. \

The password to unlock your certificate is: <u>{user\_pass}</u>. If you have any questions, please contact us at abce@psu.edu. \

Stay connected with us on <a href="https://www.linkedin.com/company/penn-state-abington-continuing-education/?viewAsMember=true ">LinkedIn</a> \

and <a href = "https://www.facebook.com/PSUAbingtonCE/">Facebook<a/>.<pre>'

                    msg.attach(MIMEText(body, 'html'))

                    msg['Subject'] = f'Certificate for completion'

                    msg['From'] = sender

                    msg['To'] = receiver

                    binary\_pdf = open(encrypted\_pdf, 'rb')

                    payload = MIMEBase('application', 'octate-stream', Name=f"{values['Name']}.pdf")

                    payload.set\_payload(binary\_pdf.read())

*# enconding the binary into base64*

                    encoders.encode\_base64(payload)

*# add header with pdf name*

                    payload.add\_header('Content-Decomposition', 'attachment', filename=f"{values['Name']}.pdf")

                    msg.attach(payload)

*with* smtplib.SMTP('smtp.office365.com', 587) *as* server:

                        server.starttls()

                        server.login(sender, password)

                        server.sendmail(sender, receiver, msg.as\_string())

*#Pop up verifying email was sent*

                        sg.popup("Certificate Has been emailed to: " + values['Email'])

    window.close()

*# define layout*

selection = [[sg.Text('Select an option to Create Certificates', size=(31, 1), font='Lucida', justification='left')],

          [sg.Combo(['Using Excel Sheet', 'Entering Details'], enable\_events=True,

                     default\_value='Using Excel Sheet', key='choice')],

          [sg.Button('SELECT', font=('Times New Roman', 12)), sg.Button('CANCEL', font=('Times New Roman', 12))]]

window = sg.Window('Certificate Generator', selection, resizable = True)

*while* True:

    event, values = window.Read()

*if* event is None or event == 'CANCEL':

*break*

*if* event == 'SELECT':

        combo = values['choice']  *# use the combo key*

*if* combo == 'Using Excel Sheet':

            generate\_from\_excel()

*# print(combo)*

*elif* combo == 'Entering Details':

            open\_window()

*# print(combo)*

window.Close()

# References

Python Libraries: <https://docs.python.org/3/library/>

Automating Mail Merge: <https://www.youtube.com/watch?v=DbPV4ccOZ8k>

PDF Encryption: <https://stackoverflow.com/questions/43475295/encrypt-pdfs-in-python>

Using PySimpleGUI: <https://www.youtube.com/watch?v=fziZXbeaegc>

MIME (Email): <https://pythonprogramming.altervista.org/send-a-pdf-with-an-email-with-python/?doing_wp_cron=1670259272.8124799728393554687500>