ASMIS - Implemented Solution

Referring to Queens Medical Centre's previous report for ASMIS, the **two-factor authentication (2FA)** was listed as one of the Cyber Security technologies. This document will provide detailed steps of the implementation of the basic 2FA algorithm in Python codes, for ASMIS.

One-time passwords (OTPs) are authentication techniques widely used as part of two-factor identification (2FA) which tend to satisfy the previously mentioned needs. OTPs are unique passwords that are only valid for a single login session for a given period of time. OTPs can be distributed to end-users via SMS that is implemented for ASMIS. And once the user's credential has been checked, the user is allowed to access the appointment schedule website. (Send OTP on Mobile using Python, 2020).

Several benefits and disadvantages can arise from using OTP as a two-factor authentication. *Table 1* shows a list of the above-mentioned (2020).

Benefits	Drawbacks
✓ Ease to use ✓ Works offline ✓ Not Vulnerable to Replay Attacks ✓ Low Cost	✘ Requires a Mobile Device ✘ Devices Can Be Lost or Stolen

Table 1 - Benefits and drawbacks of OTP

The implementation of the One-Time Password (OTP) for Queens Medical Centre relied on the following requirements:

- Flask framework
- Twilio API
- Virtual environment

Flask Framework - is a micro web framework written in Python which can migrate HTML templates to the implementation.

Twilio API - a service provider web application that interact with python, helping us to send OTP on the mobile phone, as illustrated in *figure 1*.

Virtual environment - used to manage the dependencies for the project, both in development and in production.

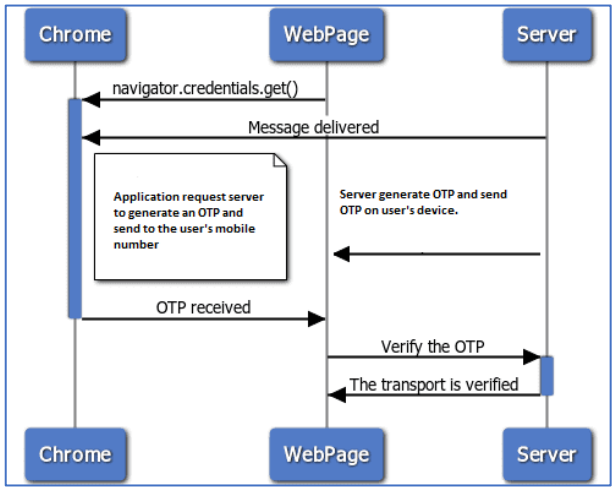

Figure 1 - Twilio Data Flow

The following steps were followed to integrate **Twilio** to **Flask**:

1. Creation of an account on https://www.twilio.com/
2. Twilio provide ***account_session_id*** and ***auth_token***
3. For python, you need to install twilio, you can do this by using
   `pip install twilio`
4. From Twilio get your ***account_sid*** and ***auth_token*** which is used by Twilio Client module like:
   `client = Client (account_sid, auth_token)`

Implementation

Below are the lines of codes in the images used in the different stages of the implementation of the Python OTP.

- ### Creating the Views

```
# importing all the classes from flask module and the class Client from twilio.rest API
from flask import *
from twilio.rest import Client
# importing packages random as it are used at line 61
import random

# the Flask constructor takes the name of current module (__name__) as argument
app = Flask(__name__)

# used for the code to be executed standalone
if __name__ == '__main__':

# the run() method of Flask class runs the application on the local development server
# host is set to 0.0.0.0 to have server available externally and port is set to 5000
    app.run('0.0.0.0', '5000')
```

Figure 2 - Creating the views

- ### Creating the Template for Login

```html
<html>

<head>
    <title>  Queens Medical Centre - Login </title>
 </head>
<body>
   <!-- form with action /getOTP and post method-->
    <form action='/getOTP' method="POST">
        <div>
            <h2> Queens Medical Centre - Login </h2>
          <!-- textbox for username -->
            <label> Username Name: </label>
            <input type="text" name="uname" required/><br> <br>
          <!-- textbox for password -->
            <label> Password: </label>
            <input type="password" name="passwrd" required/><br> <br>
          <!-- textbox for mobile phone number -->
            <label> Mobile Number: </label>
            <input type="text" name="number" placeholder="+230 * * *" required/> <br> <br>
          <!-- login form submission button -->
            <button type="submit"> Login </button>
        </div>
     </form>

 </body>
 </html>
```

Figure 3 - Creating the Template for Login

Creating the Template for Validating OTP

```html
<html>

<head>
    <title> Validate OTP (One Time Password)</title>
 </head>
<body>
   <!-- form with action /validateOTP and post method -->
    <form action='/validateOTP' method="POST">
        <div>
            <h2> Queens Medical Centre - Validate OTP </h2>

            <label>A  OTP (One Time Password) has been sent to your number. </label> <br>
            <label>Please enter the OTP in the field below to verify your user profile. </label> <br> <br>
          <!-- textbox to insert otp received by sms-->
            <input type="text" name="otp" required/>
          <!-- enterOTP form submission button -->
            <button type="submit"> Validate OTP </button>
        </div>
    </form>

 </body>
 </html>
```

Figure 4 - Creating the Template for Validating OTP

Setting up Twilio

```python
    def getOTPApi(number):
    # account_sid and auth_token obtained from Twilio API
          account_sid = 'AC58b94c8d51a9a93e406cd5beb173a228'
          auth_token = '036c206e147db92ee0f71933a82d58c1'

        # passing the account_sid and auth_token as parameters in Twilio's Client class
          client = Client(account_sid, auth_token)
          otp = generateOTP() # calling generateOTP function to get unique OTP

          # session allows the storage of information that is specific to a user from one request to the next
          session['response'] = str(otp)

          # message to be sent to user along with generated otp
          body = 'Your OTP is: ' + str(otp)
          message = client.messages.create(
              from_='+12283357039', # virtual phone number obtained from Twilio
              body =body,
              to = number
      )
          # sending message to specified mobile phone number
          if message.sid:
              return True
```

Figure 5 - Setting up Twilio

- ## Generating the OTP

```
# function generateOTP is used to randomly generate 6 digits value
def generateOTP():
    return random.randrange(100000,999999)

```

Figure 6 - Setting up Twilio

- ## Validating the OTP

```
# function validateOTP is used to check whether the inserted otp in form is the same as otp received by sms
def validateOTP():
  # get the otp value inserted in form
    otp = request.form['otp']

    # using session for otp
    if 'response' in session:
        s = session['response']
        # terminate session
        session.pop('response', None)

        # inserted otp is same as otp received by sms
        if s == otp:
            return 'OTP is correct. Successful log in'
        # inserted otp is different from otp received by sms
        else:
            return 'OTP is incorrect. Unsuccessful log in'
```

Figure 7 - Validating the OTP

Testing

1. Executing the Python script opens the login.html page, as illustrated in *figure 8*.

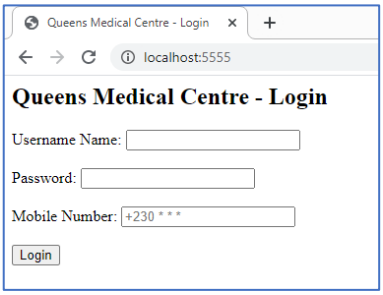

Figure 8 - Login Form

2. Filling in all the details in login form as illustrated in *figure 9*.

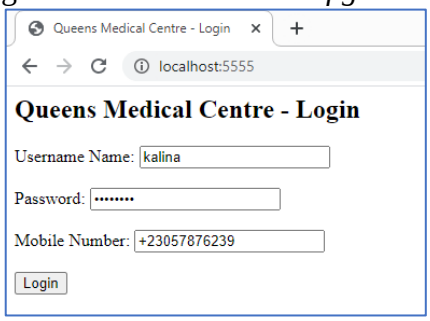

Figure 9 - Filling details in login form

3. Login in which redirects to enterOTP.html as illustrated in *figure 10*.

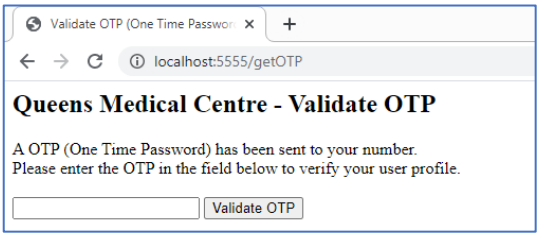

Figure 10 - enterOTP page

4. OTP obtained through SMS on mobile phone, as illustrated in *figure 11*.

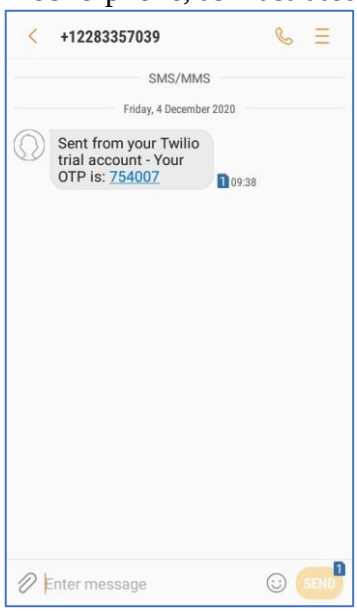

Figure 11 - OTP received through SMS

5. Entering correct OTP (*figure 12*) – confirmation of successful login obtained, as illustrated in *figure 13*.

Figure 12 - Entering correct OTP

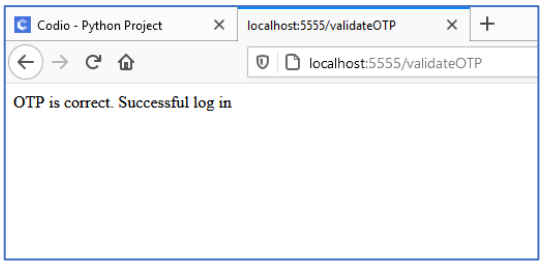

Figure 13 - Successful login

6. Entering correct OTP (*figure 14*) – error message of unsuccessful login obtained, as illustrated in *figure 15*.

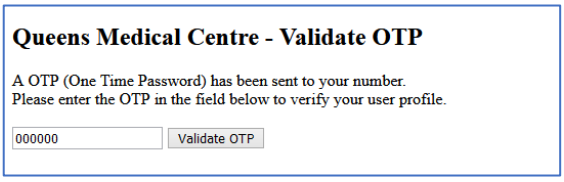

Figure 14 - Entering incorrect OTP

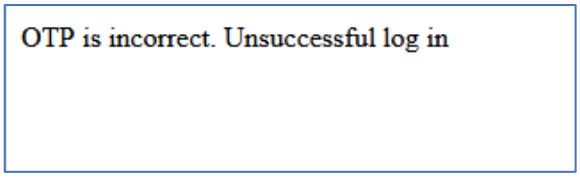

Figure 15 - Unsuccessful login

References:

Medium. 2020. Send OTP On Mobile Using Python. [online] Available at: <https://prataplyf.medium.com/send-otp-on-mobile-using-python-491a472853a9> [Accessed 4 December 2020].

2020. [online] Available at: <https://blog.identitya utomation.com/two-factor-authentication-2fa-explained-one-time-password-soft-tokens> [Accessed 4 December 2020].

Appendix:

Command lines used to run the ASMIS web application:

```
1. codio@greek-ruby:~/workspace$ cd asmis
2. codio@greek-ruby:~/workspace$ virtualenv app
3. codio@greek-ruby:~/workspace$ source app/bin/activate
4. (app) codio@greek-ruby:~/workspace/asmis$ cd app
5. (app) codio@greek-ruby:~/workspace/asmis$ install flask
6. (app) codio@greek-ruby:~/workspace/asmis$ install twilio
7. (app) codio@greek-ruby:~/workspace/asmis/app$ python script.py
```

Phone number registration on Twilio:

- Request to add required phone number to Twilio by email