SUMMATIVE ASSESSMENT 2

PROGRAMMING LAB- Group Project

Project Name: Secure Login

Group Members: Ashiq, Fahad, & Salman

Introduction and Background

Today in this era of technology, secure login has become one of the main properties of utmost necessity. Secure login as a process has become one of the most crucial components of online security, which involves checking and verifying a user before the user can access the system or service. Secure login is vital in securing confidential information and making sure to prevent unauthorized access. The integrity of internet security has been weakened by scams such as phishing schemes and password theft, despite the importance of secure login.

Secure login has been a large part of software engineering in developing software. One-time password authentication or in other words (OTP) authentication is most common and one of the most sophisticated authentication techniques that have arisen because of advancements in this area of technology. This is basically two factor authentication which creates a different password for each login attempt which adds an extra level of security that makes it challenging for any hackers trying to access the user's account.

Mobile phone usage for OTP-based two-factor authentication is now highly recognized and accepted as a secure login method. Again, the mobile phone produces a distinct OTP for each login attempt, which is then transmitted to the user's phone and is only effective for a short period of time(Eldefrawy et.al,2011).Using this tactic, cybercriminals will have hard time to access the particular user's account.

It is impossible for us to emphasize how important secure login is to us for online security. Although, software engineering has been an important factor which has helped in the creation of complex authentication techniques such as OTP based two factor authentication, it is also imperative for us to be aware and vigilant in developing and implementing secure login methods as technology evolves and more cybercriminals will be able to use that to their advantage but we will also improve this technique against these emerging security threats.

Problem Statement

Online security still faces several vulnerabilities that needs immediate attention despite having significant developments in secure login tactics. The widespread usage of passwords which are vulnerable and poor password hygiene among the users are the major problems. According to a recent survey by Splashdata (2021), the use of common passwords such as “123456” and “passwords” shows that users are not adequately aware of the importance of strong passwords. Due to this, many websites require users to establish challenging passwords that would be incredibly difficult for them to remember and store in a database of passwords. This requires employing strong authentication installing strong authentication methods that ensures secure login. According to a study made by Verizon (2021), phishing is one of the most prominent cause of data intrusions in 2020, making it the most prevalent causes of intrusions. A validated option that adds an extra step towards security decreases the risks associated with using insecure passwords is two-factor authentication(2FA). The usage of two factor authentication can stop unauthorized access and protect confidential information from phishing scams. To address these issues and increase online security, it is important to implement and enhance secure login methods like 2FA.

Aims & Objectives

The following article will be an elaborative case study for implementing two-factor authentication into codes that are designed to create high-security login systems. In an age where everything has become electric and all our liquid assets and data are stored online, it is necessary to improve secure login systems that can easily be implemented. In summary, the article will accomplish two tasks:

1. Analysing studies that introduce advanced ways for security login systems, most specifically, two-factor authentication. Moreover, the article will go through two major studies and review the many ways security login codes can be innovated through more advanced security systems.
2. The implementation, methodology, and discussion of a simplified code showcasing two-factor authentication like sending passcodes through emails and QR scanning systems like the Google authenticator. Furthermore, the article will elaborate on the code in ways that it imports packages and how it runs on the console.

Literature Review

There have been studies conducted over the past years that introduce new and better ways of online security in attempts to counter security threats in areas such as banks, governmental applications, the healthcare industry, military organizations, educational institutions, etc. This is where two-factor authentication comes in. Two-factor authentication (2FA) is a stronger and more secure security process that requires two forms of authentication, typically something the user knows (such as a password or PIN) and something the user possesses (such as a physical token or mobile phone), in order to access an account or system. Withdrawing money from an ATM machine is an example of 2FA, where the user must possess an ATM card and know a unique PIN. The purpose of 2FA is to add an additional layer of security to the login process, making it harder for unauthorized users to access sensitive information or systems, and reducing the likelihood of guessing or stealing both authentication factors (Schneier, 2005).

The early (2FA) systems were the ‘Tokens’ or ‘Cryptographic Tokens’, which are security devices or applications that generate unique codes that are used along with a password or PIN to authenticate a user. Although cryptographic tokens’ were implemented in both commercial and federal banks across the world, these devices have been criticised for being extremely costly. According to a research article by Jani and Patel (2016), the cost of cryptographic tokens can also be affected by factors such as production volume, the technology used, and vendor pricing policies.

A mobile-based software token that will relieve organisations of the expense of acquiring and maintaining the hardware tokens has been proposed as an alternative in a piece by Fadi Aloul & El-Hajj (2011). Additionally, buyers will be able to install several software tokens on their mobile devices. As a result, they are unlikely to worry about an assortment of hardware tokens and will simply concentrate on only their smartphones. One of the studies they present is a system in which a one-time password (OTP) is generated by the server and delivered via text message to the user's mobile device. The protocol utilises a challenge-response method in which the user responds to a challenge by inputting the OTP that was delivered to their mobile device by the server. The user's mobile device receives a text message containing the OTP after the server generates it. Another system proposed by Fadi Aloul & El-Hajj (2011), is a connection-less authentication system. To examine further, the OTP is delivered to the mobile phone using an algorithm. The algorithm is as follows:

- The OTP is generated using several factors, including the International Mobile Equipment Identity number, International Mobile Subscriber Identity number, username, pin, time, and date. The telecommunication company synchronises the time between the server and the device so that the OTP delivery process begins. These factors are combined and hashed using SHA-256, which is a secure cryptographic hash function that takes an input and returns a 256-bit message. To duplicate the PIN to the length of the 256-bit message obtained from the hash, the PIN is repeated multiple times and XOR-ed with the hash. The resulting 256-character message is then Base64 encoded, resulting in a 28-character message that is sent to the user’s mobile device. To ensure that the OTP has a fixed length specified by the administrator, the message is divided into two parts, and the first 14 characters are XOR-ed with the last 14 characters. Finally, the user receives their OTP for a short amount of time.

To explain further the algorithm used in generating the OTP, SHA-256, as Secure Hash Algorithm 256-bit, is a hash function that programmers implement in security systems due to its popularity and secure features. It receives any size of input data and performs a complex series of mathematical operations to get a unique hash value of 256 bits. The 256-bit hash value acts as a digital signature for the original message, thus any changes to the message would end up resulting in a unique hash value, demonstrating the tampering (Gueron, Johnson, and Walker, 2011).

With the OTP algorithm, a Java 2 Micro Edition is installed into the mobile device that has an easy-to-use GUI for the user to run the OTP program. After the user enters their username and pin, they will have to decide which OTP generation method they would like to use to submit to the server for verification. They are as follows:

- Connection-Less Authentication System: a system where a one-time password is generated on a mobile phone without needing to connect to a server. The phone acts as a token and generates a unique password using certain factors, and the server has the same factors to verify the password submitted by the client. A program on the mobile phone generates the one-time password.

- SMS-Based Authentication System: a system where the mobile phone can request a one-time password from the server directly. The mobile phone sends unique user information to the server via an SMS message, and if the information is correct, the server sends a randomly generated OTP back to the mobile phone. Recommended if the former is unsuccessful.

Finally, a database is required for the server to store the user's factors and a server is required for the OTP for the organization (such as a company or website that sets up a computer server) to generate the OTP. The system proposed by Fadi Aloul & El-Hajj (2011) was successfully implemented into devices such as Nokia E60 and Nokia E61 and the results are remarkable. Each user would have one lack OTPs generated-all of them unique from one another.

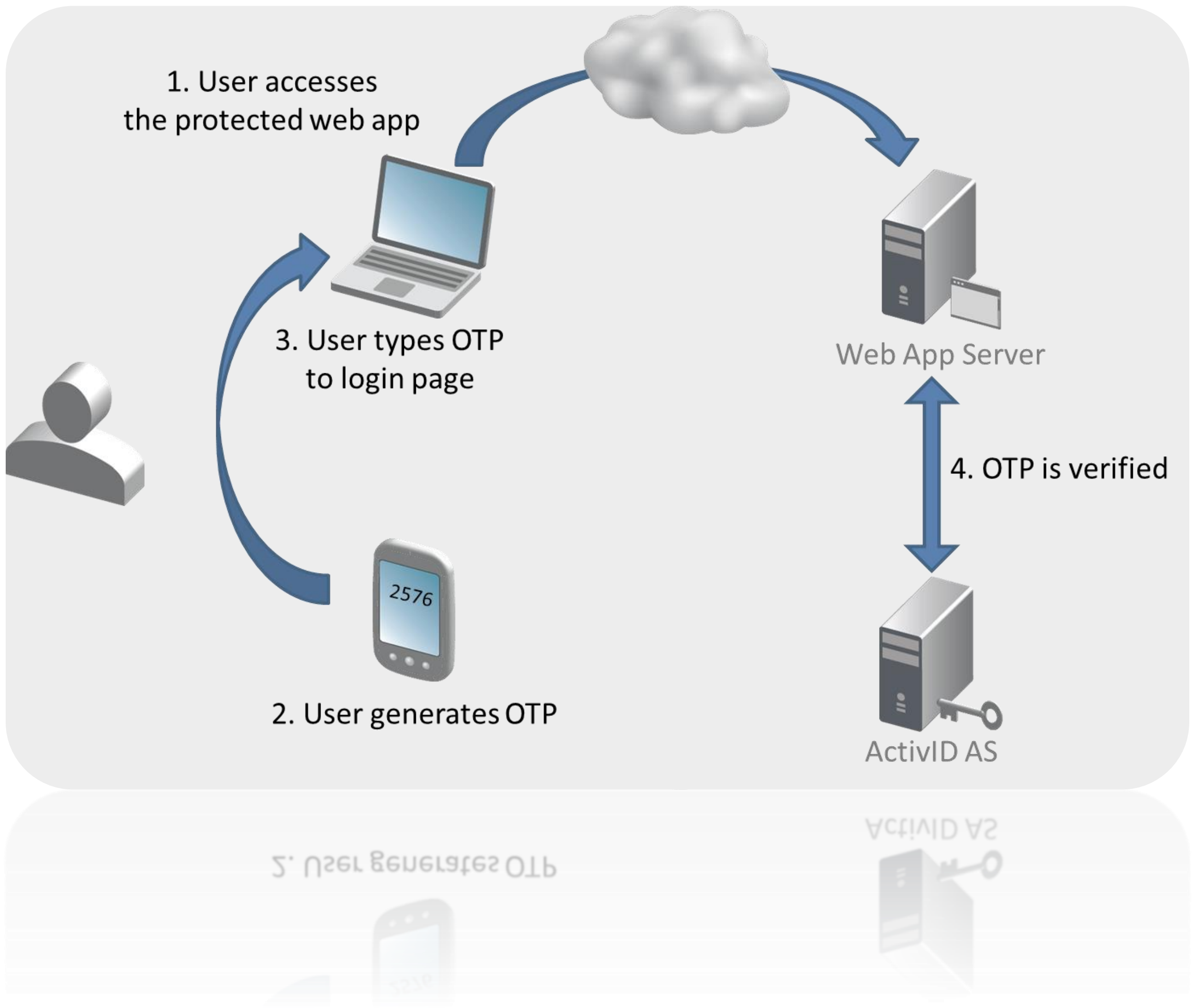
Conversely, a study by H.Eldefrawy, Alghathbar, and K.Khan (2011) introduced a new way of generating the OTP.

Their approach involves generating a one-time password (OTP) on the user's mobile phone using a mobile application and then requiring the user to enter the OTP along with their regular password to log in to the system. The OTP is generated using a combination of a secret key that is shared between the server and the mobile application, and a counter that is incremented each time an OTP is generated. Furthermore, their scheme also implements the SHA-256 algorithm to generate the OTP. They utilised SHA-256 for generating the intermediate value int s required to create the session OTP. According to the authors, SHA-256 is a robust hash function that is resistant to numerous forms of attacks, making it appropriate for use in their OTP generation system. The login and authentication process are as follows (Figure 1 provides a brief illustration):

- The user logins into a service provider's website and requests access. Then a secure session is established where the user can enter their own username and password thus the first-factor authentication. The user also provides the current status that allows the server to synchronize its seed with the user's current seed to insure that both

sides hold the same seed value before the sending challenge.
- New indexes are shown to the user at random by the server. For the purpose of getting the corresponding OTP, the user inputs those indexes into his OTP generator.
- The user answers by providing the correct OTP. The server contrasts the estimated OTP with its counterpart that was received.
- Finally, an approval execution or communication termination will be transferred by the server.

Figure 1.

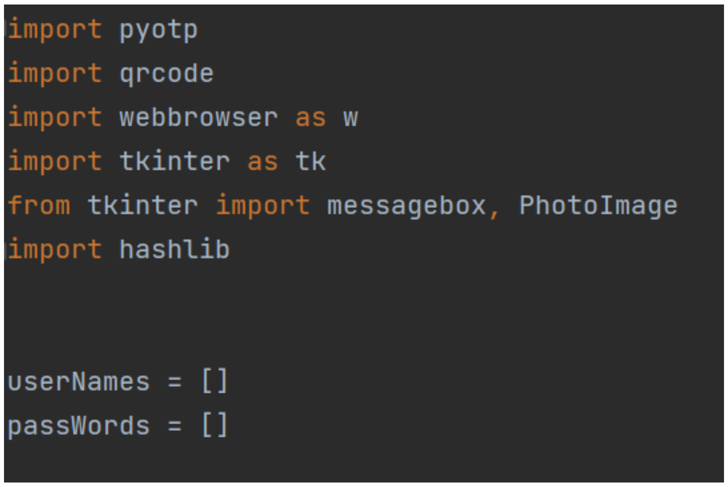

1. User accesses the protected web app
3. User types OTP to login page
Web App Server
4. OTP is verified
2576
2. User generates OTP
ActivID AS

Authors H.Eldefrawy, Alghathbar, and K.Khan (2011) accord a brief security analysis of their advocated system. According to their analysis, the system is able to block any possible attacks and can prevent hackers from impersonating either the user or the service provider. For instance, if a hacker takes control of the user’s device and tries to access their account, the service provider can challenge the user in such a way that it would be impossible for the attacker to predict the next challenge thus preventing theft. Another instance of the proposed system's tight security occurs if an attacker attempts to create an OTP matching a given challenge yet is unable to do so due to the fact they lack the information of the value of int s, which must be entered to properly update the session OTP for acceptance by the host. As a whole, the prosed system showcases that any form of cyber-crimes is impractical.

To sum up, 2FA protocols that use mobile devices as a second factor thus offer a new way of enhancing cyber-security. Examining the two case studies invokes various possibilities for the implementation of either one or the other in the future.

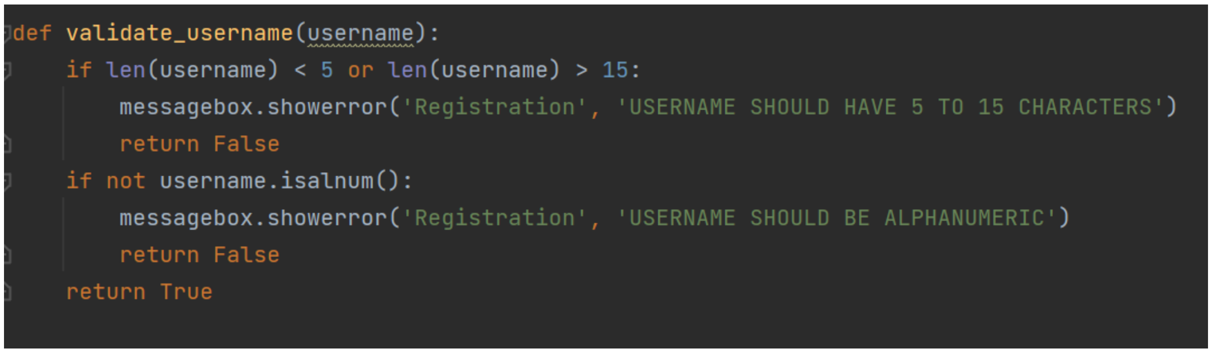
Methods & Implementation & Discussion:

Our system is a secure login system implemented in Python to access the official Royal Holloway website. The code of our system starts by importing the required libraries/packages such as pyotp, qrcode, tkinter, webbrowser, and hashlib. Then, it initializes two empty lists 'userNames' and 'passWords' to store the username and password of registered users.

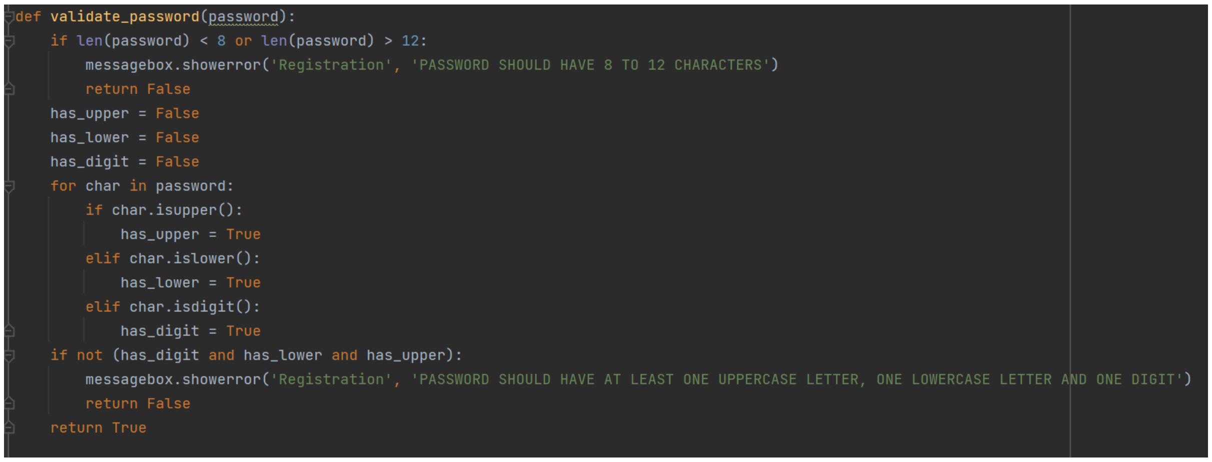


```
import pyotp
import qrcode
import webbrowser as w
import tkinter as tk
from tkinter import messagebox, PhotoImage
import hashlib


userNames = []
passWords = []
```

The 'register()' function is defined to register a new user with a unique username and password. Firstly, it checks for blank spaces meaning if a user has entered something or not in the input fields and displays error message accordingly then the username and password entered by the user are passed through validation checks ('validate_username()' and 'validate_password()' functions) to make sure that the user is following the criteria for a strong username and a stronger password. The criteria have been defined in the validation functions and is as follows:



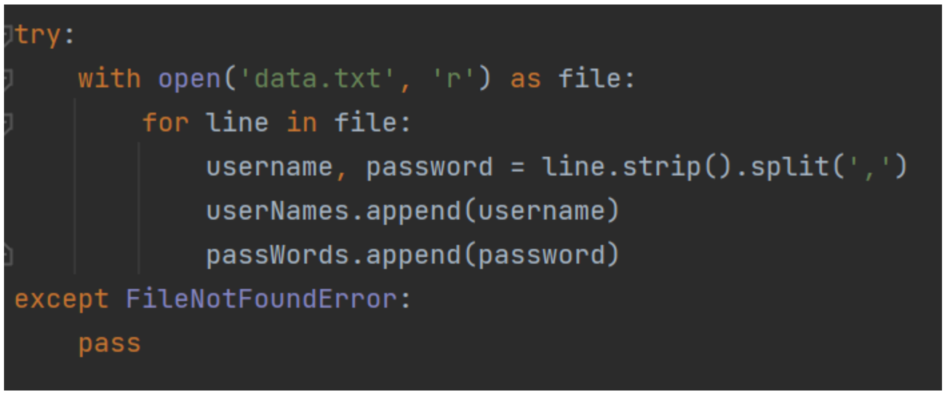
```python
def validate_username(username):
    if len(username) < 5 or len(username) > 15:
        messagebox.showerror('Registration', 'USERNAME SHOULD HAVE 5 TO 15 CHARACTERS')
        return False
    if not username.isalnum():
        messagebox.showerror('Registration', 'USERNAME SHOULD BE ALPHANUMERIC')
        return False
    return True
```



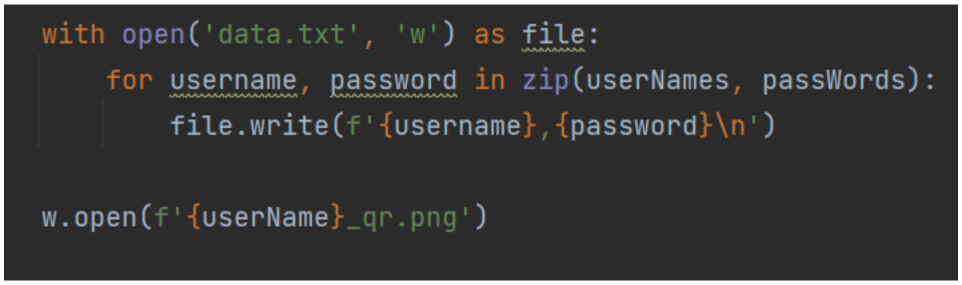
```python
def validate_password(password):
    if len(password) < 8 or len(password) > 12:
        messagebox.showerror('Registration', 'PASSWORD SHOULD HAVE 8 TO 12 CHARACTERS')
        return False
    has_upper = False
    has_lower = False
    has_digit = False
    for char in password:
        if char.isupper():
            has_upper = True
        elif char.islower():
            has_lower = True
        elif char.isdigit():
            has_digit = True
    if not (has_digit and has_lower and has_upper):
        messagebox.showerror('Registration', 'PASSWORD SHOULD HAVE AT LEAST ONE UPPERCASE LETTER, ONE LOWERCASE LETTER AND ONE DIGIT')
        return False
    return True
```

- The username should be 5 to 15 characters long and it should be alphanumeric.
- The password should be 8 to 12 characters long.
- The password should have at least one uppercase letter, one lowercase letter, and one digit.

Furthermore, it checks if the username is already taken, and if not, it adds the new user’s username and hashed password (talked about later) to the respective lists. The secret key (talked about later) is also generated in this function and saved to a text file with user’s username appended to it. Finally, the username and hashed passwords are saved to a file named ‘data.txt’ where they are stored permanently (second image below) so that every time the code is rerun, it reads from this file (first image below) and adds the already registered accounts to the lists.

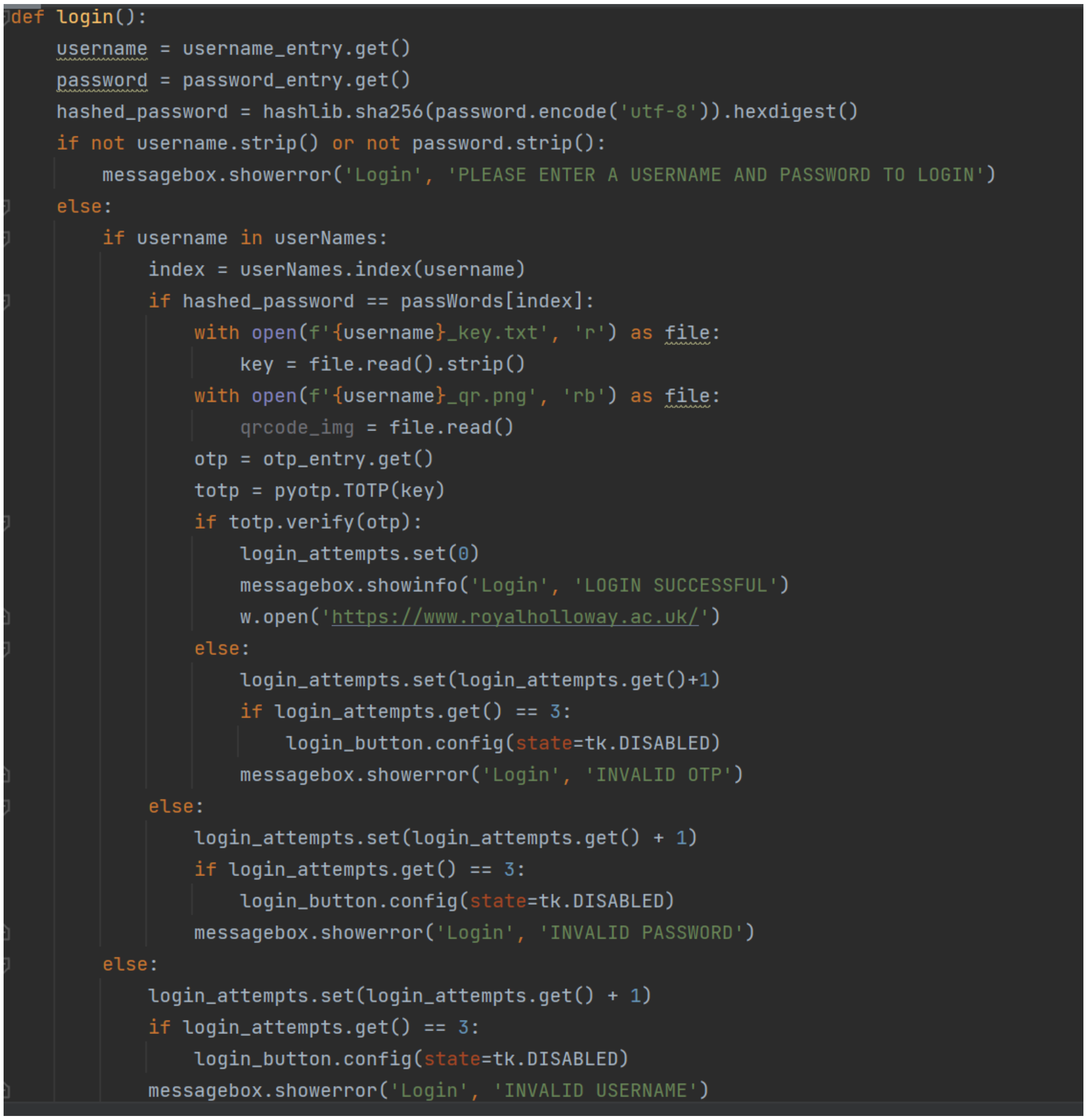


```
try:
    with open('data.txt', 'r') as file:
        for line in file:
            username, password = line.strip().split(',')
            userNames.append(username)
            passWords.append(password)
except FileNotFoundError:
    pass
```



```
with open('data.txt', 'w') as file:
    for username, password in zip(userNames, passWords):
        file.write(f'{username},{password}\n')

w.open(f'{userName}_qr.png')
```

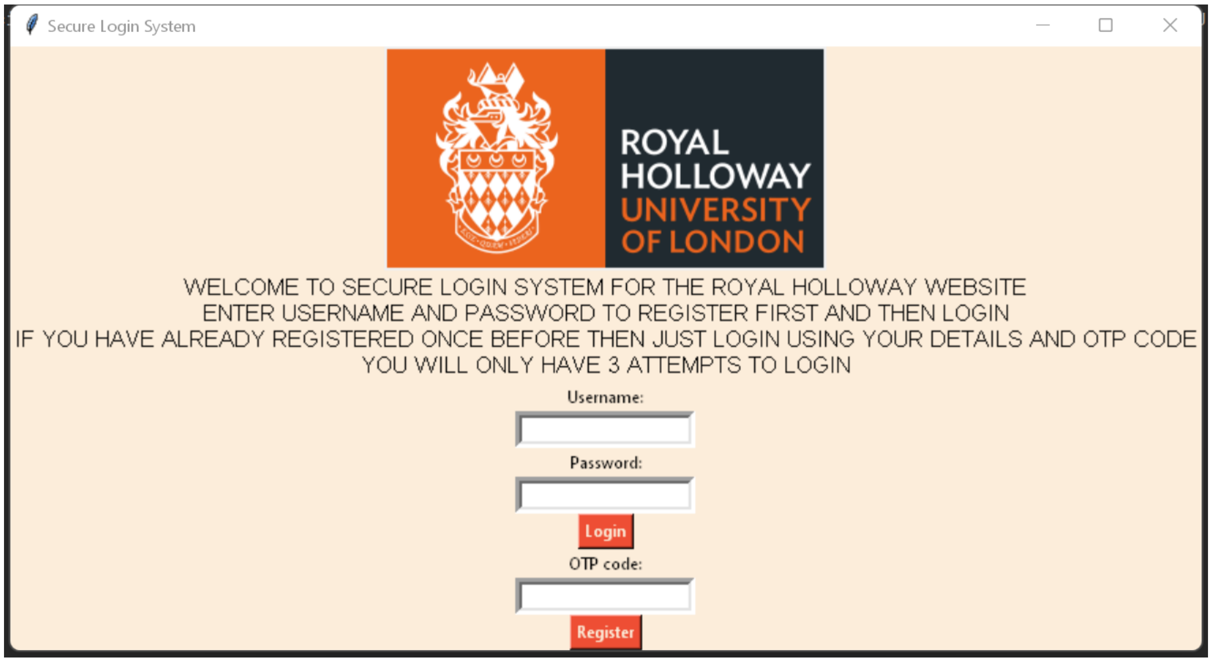
The ’login()’ function is defined to log in an already registered user. It takes the username and password from the input fields and checks if the username and password match those stored in the ‘userNames’ and ‘passWords’ lists. If they match, it reads the key from the file with the user’s username appended to it and then prompts the user to enter the OTP generated on their mobile device. The three login attempts implementation has also been done in this function.



```python
def login():
    username = username_entry.get()
    password = password_entry.get()
    hashed_password = hashlib.sha256(password.encode('utf-8')).hexdigest()
    if not username.strip() or not password.strip():
        messagebox.showerror('Login', 'PLEASE ENTER A USERNAME AND PASSWORD TO LOGIN')
    else:
        if username in userNames:
            index = userNames.index(username)
            if hashed_password == passWords[index]:
                with open(f'{username}_key.txt', 'r') as file:
                    key = file.read().strip()
                with open(f'{username}_qr.png', 'rb') as file:
                    qrcode_img = file.read()
                otp = otp_entry.get()
                totp = pyotp.TOTP(key)
                if totp.verify(otp):
                    login_attempts.set(0)
                    messagebox.showinfo('Login', 'LOGIN SUCCESSFUL')
                    w.open('https://www.royalholloway.ac.uk/')
                else:
                    login_attempts.set(login_attempts.get()+1)
                    if login_attempts.get() == 3:
                        login_button.config(state=tk.DISABLED)
                    messagebox.showerror('Login', 'INVALID OTP')
            else:
                login_attempts.set(login_attempts.get() + 1)
                if login_attempts.get() == 3:
                    login_button.config(state=tk.DISABLED)
                messagebox.showerror('Login', 'INVALID PASSWORD')
        else:
            login_attempts.set(login_attempts.get() + 1)
            if login_attempts.get() == 3:
                login_button.config(state=tk.DISABLED)
            messagebox.showerror('Login', 'INVALID USERNAME')
```

GUI

The main window which is opened after executing the code is created using the tkinter library. It displays the Royal Holloway logo, the instructions for the registration and login process, and the input fields for everything. It also has two buttons ‘Login’ and ‘Register’, to initiate the respective actions, both buttons are linked to their respective functions. The overall look of the window has also been designed in tkinter.


Secure Login System

WELCOME TO SECURE LOGIN SYSTEM FOR THE ROYAL HOLLOWAY WEBSITE
ENTER USERNAME AND PASSWORD TO REGISTER FIRST AND THEN LOGIN
IF YOU HAVE ALREADY REGISTERED ONCE BEFORE THEN JUST LOGIN USING YOUR DETAILS AND OTP CODE
YOU WILL ONLY HAVE 3 ATTEMPTS TO LOGIN

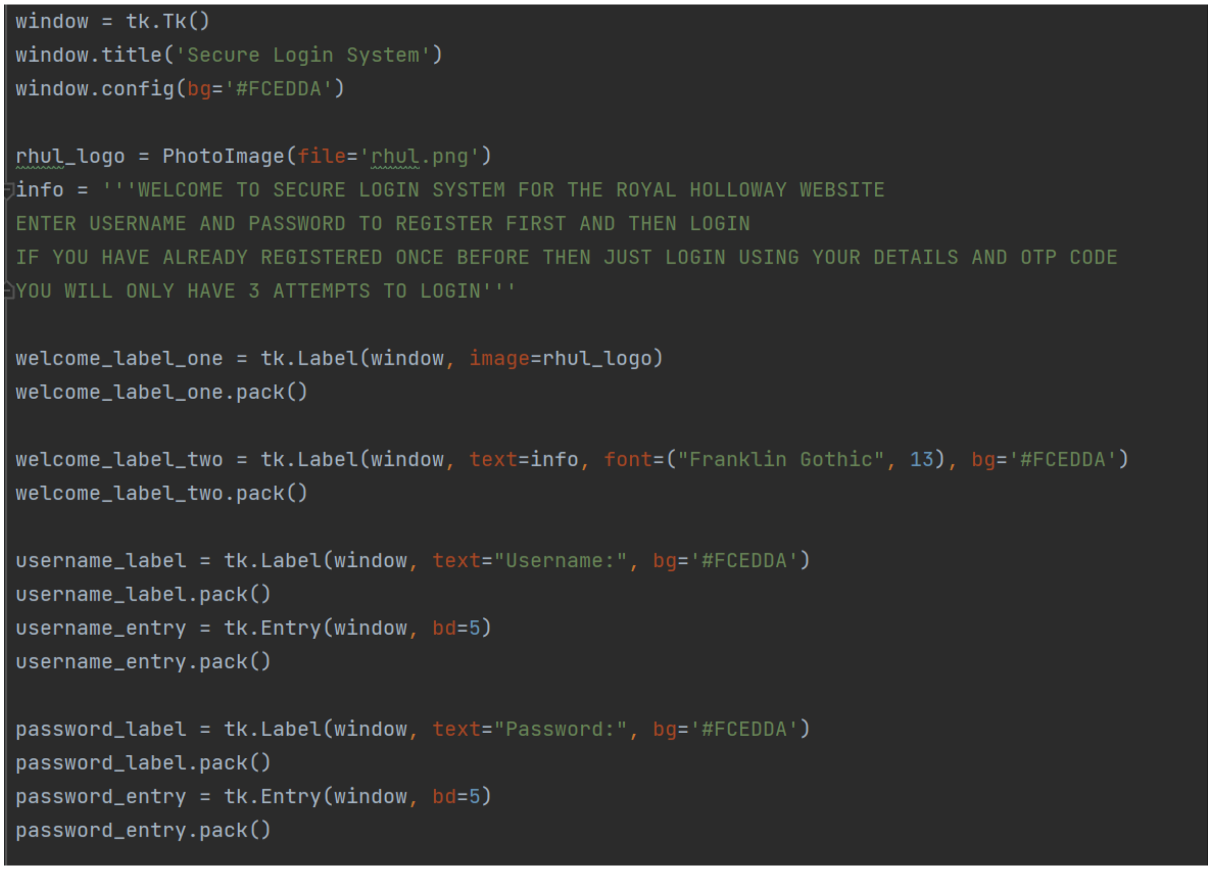
Username:

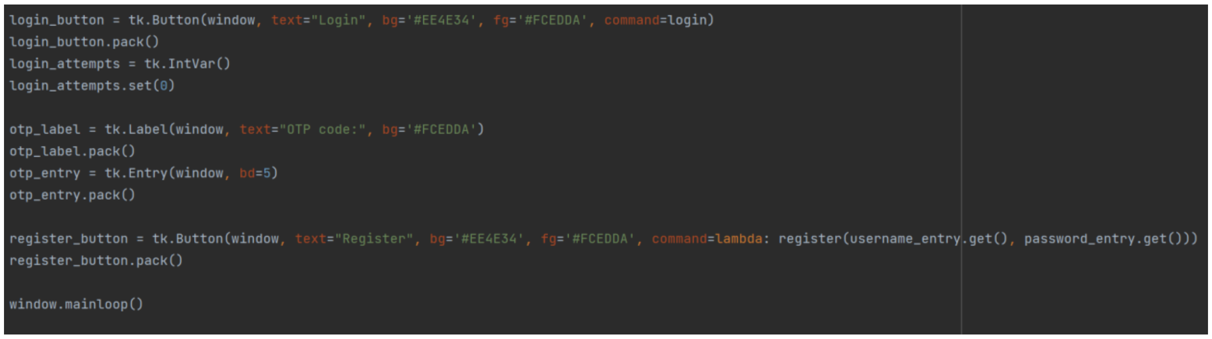
Password:

Login

OTP code:

Register


```
window = tk.Tk()
window.title('Secure Login System')
window.config(bg='#FCEDDA')

rhul_logo = PhotoImage(file='rhul.png')
info = '''WELCOME TO SECURE LOGIN SYSTEM FOR THE ROYAL HOLLOWAY WEBSITE
ENTER USERNAME AND PASSWORD TO REGISTER FIRST AND THEN LOGIN
IF YOU HAVE ALREADY REGISTERED ONCE BEFORE THEN JUST LOGIN USING YOUR DETAILS AND OTP CODE
YOU WILL ONLY HAVE 3 ATTEMPTS TO LOGIN'''

welcome_label_one = tk.Label(window, image=rhul_logo)
welcome_label_one.pack()

welcome_label_two = tk.Label(window, text=info, font=("Franklin Gothic", 13), bg='#FCEDDA')
welcome_label_two.pack()

username_label = tk.Label(window, text="Username:", bg='#FCEDDA')
username_label.pack()
username_entry = tk.Entry(window, bd=5)
username_entry.pack()

password_label = tk.Label(window, text="Password:", bg='#FCEDDA')
password_label.pack()
password_entry = tk.Entry(window, bd=5)
password_entry.pack()
```



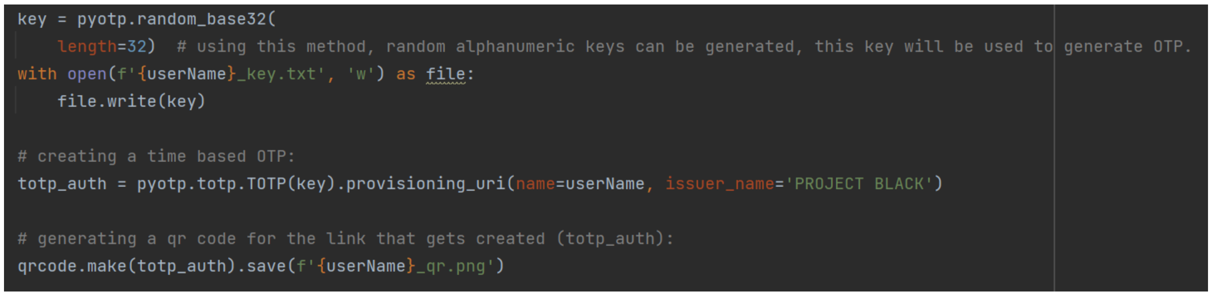
```
login_button = tk.Button(window, text="Login", bg='#EE4E34', fg='#FCEDDA', command=login)
login_button.pack()
login_attempts = tk.IntVar()
login_attempts.set(0)

otp_label = tk.Label(window, text="OTP code:", bg='#FCEDDA')
otp_label.pack()
otp_entry = tk.Entry(window, bd=5)
otp_entry.pack()

register_button = tk.Button(window, text="Register", bg='#EE4E34', fg='#FCEDDA', command=lambda: register(username_entry.get(), password_entry.get()))
register_button.pack()

window.mainloop()
```

Two-Factor Authentication (2FA)

In this section, we discuss the implementation of two-factor authentication in our secure login system. To implement 2FA, we utilized the pyotp library, which provides functionality for generating and validating OTPs. Our implementation follows the time-based OTP (TOTP) approach, where the OTP is based on a shared secret key which is generated by the pyotp library, and it changes every few seconds. This is done in the ‘register()’ function.

OTP Generation and Provisioning

When a user registers for an account, we generate a random 32-character secret key using the **‘pyotp.random_base32’** function. This key is unique to each user and stored securely in a separate text file for future use. To provision the OTP to the user’s mobile device, we generate a provisioning URI using the TOTP class from pyotp. This URI contains the user’s name, issuer name (e.g., “PROJECT BLACK”), and the secret key. We use the qrcode library to convert this URI into a QR code image. The user can then scan this QR code using any compatible authenticator app, such as Google Authenticator.



```
key = pyotp.random_base32(
    length=32)  # using this method, random alphanumeric keys can be generated, this key will be used to generate OTP.
with open(f'{userName}_key.txt', 'w') as file:
    file.write(key)

# creating a time based OTP:
totp_auth = pyotp.totp.TOTP(key).provisioning_uri(name=userName, issuer_name='PROJECT BLACK')

# generating a qr code for the link that gets created (totp_auth):
qrcode.make(totp_auth).save(f'{userName}_qr.png')
```

Login Process

During the login process, the user provides their username, password, and OTP. We first verify the username and password against our stored credentials. If they match, we retrieve the secret key associated with the user from the text file. Using this key, we create an instance of the TOTP class from pyotp then we compare both the provided OTP against the OTP generated by the TOTP instance at the current time. If the OTPs match, the user is granted access to the Royal Holloway website by using the webbrowser library and the login process is successful. If the OTPs do not match, we display appropriate error messages to the user.

User Experience and Security

To enhance the user experience, we have provided clear instructions which need to be followed during the registration and login processes and we also display informative messages to guide the users throughout. For the best user experience, we have also implemented Graphical User Interface (GUI) using the tkinter library. In terms of security, the secret keys are stored securely in a separate text file and the passwords are hashed (using hashlib library) using the SHA-256 hashing algorithm. The user also has only three attempts to login and after three failed attempts, the Login button is disabled for use.

Discussion

Overall, the code implements a secure and effective login and registration system with two-factor authentication. The system ensures only registered users with the correct username, password and OTP can access the Royal Holloway website. The code is written in a modular way, making it easy to understand and modify. However, there are a few areas where the code can be improved in terms of security and perfection. For example, the code can be improved by using an actual database instead of files and lists to store user data.

Conclusion

This article accomplished in with its three main goals. The first was to clarify the idea of secure login systems and discuss the problems with insufficient security. The second involves comparing two research that look at two different approaches to creating the one-time password. Finally, we presented a code which employs the same principles and mechanics as the two studies to develop amore straightforward code. This code would use a disconnecting system to generate an OTP by scanning a QR code and using the SHA-256 algorithm, and it would use the user's device to generate a passcode that would grant permission to access to the service provider, In this case, the website if Royal Holloway. In the paper, the importance if a new two-factor authentication system is highlighted.

References

Aloul, F., Zahidi, S., & El-Hajj, W. (2011). Two Factor Authentication Using Mobile Phones. Journal of Information Security, 2(2), 105-113.

Eldefrawy, M.H., Alghathbar, K. and Khan, M.K, (2014). OTP-Based Two-Factor Authentication Using Mobile Phones. Journal of Computer Networks and Communications.

Jani, N. N., & Patel, D. D. (2016). Analysis of the Cryptographic Token's Cost and

Its Implementation in the Security System. 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET) (pp. 2304-2309).

Schneier, B. (2005). Two-Factor Authentication: Too Little, Too Late. Inside Risks 178, Communications of the ACM, 48(4), April 2005.

SplashData. (2021). Worst Passwords List of 2021. Retrieved from https://www.teamsid.com/worst-passwords-list/

Verizon. (2021). 2021 Data Breach Investigations Report: Summary of Findings. Retrieved from https://enterprise.verizon.com/resources/reports/dbir/2021/summary-of-findings/

Gueron, S., Johnson, S., & Walker, J. (2011). "SHA-512/256". In 2011 Eighth International Conference on Information Technology: New Generations (pp. 354-358). Las Vegas, NV, USA: IEEE. doi: 10.1109/ITNG.2011.69.