ONE-TIME PASSWORD VERIFICATION SYSTEM

A Project Report

Submitted in the partial fulfillment of the requirements for the

award of the degree of

Bachelor of Technology

in

Department of Computer Science and Engineering

By

TEJASWI KOTTAKKI	(2010030088)
HARIKA NETHI	(2010030119)
SHREYA CHINTAWAR	(2010030155)
AMRUTHA VARSHINI	(2010030009)

under the supervision of

CHANDA RAJ KUMAR RAO

PROFESSOR

Department of Computer Science and Engineering

K L University Hyderabad,

Aziz Nagar, Moinabad Road, Hyderabad – 500 075, Telangana, India.

March, 2022

Declaration

The Project Report entitled "ONE-TIME PASSWORD VERIFICATION SYSTEM" is a record of bonafide work of Tejaswi Kottakki (2010030088), Harika Nethi (2010030119), Shreya Chintawar (2010030155), Amrutha Varshini (2010030009), submitted in partial fulfillment for the award of B.Tech in the Department of Computer Science and Engineering to the K L University, Hyderabad. The results embodied in this report have not been copied from any other Departments/University/Institute.

<Signature of the Students >

Certificate

This is to certify that the Project Report entitled "ONE-TIME PASSWORD VERIFICATION SYSTEM" is being submitted by Tejaswi Kottakki (2010030088), Harika Nethi (2010030119), Shreya Chintawar (2010030155), Amrutha Varshini (2010030009) submitted in partial fulfillment for the award of B.Tech in Computer Science to the K L University, Hyderabad is a record of bonafide work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute.

Signature of the Supervisor

Name and Designation

Signature of the HOD

Signature of the External Examine

ACKNOWLEDGEMENT

First and foremost, we thank the lord almighty for all his grace & mercy showered upon us, for completing this project successfully.

We take grateful opportunity to thank our beloved Founder and Chairman who has given constant encouragement during our course and motivated us to do this project. We are grateful to our Principal **Dr. L. Koteswara Rao** who has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgement & sincere thanks to our Head of the Department **Dr. Chiranjeevi Manike** for her exemplary guidance, monitoring and constant encouragement throughout the course of the project. We thank **Chanda Raj Kumar Rao** of our department who has supported throughout this project holding a position of supervisor.

We whole heartedly thank all the teaching and non-teaching staff of our department without whom we won’t have made this project a reality. We would like to extend our sincere thanks especially to our parent, our family members and friends who have supported us to make this project a grand success.

ABSTRACT

This document describes an extension of One-Time Password (OTP) algorithm, namely the HMAC based One Time Password (HOTP) algorithm to support the time-based moving factor. The HOTP algorithm specifies an event-based OTP algorithm, where the moving factor is an event counter. The present work bases the moving factor on a time value. A time-based variant of the OTP algorithm provides short-lived OTP values, which are desirable for enhanced security.

The proposed algorithm can be used across a wide range of network applications, from remote Virtual Private Network (VPN) access and Wi-Fi network logon to transaction-oriented Web applications. The authors believe that a common and shared algorithm will facilitate adoption of two-factor authentication on the Internet by enabling interoperability across commercial and open-source implementations

TABLE OF CONTENTS

1. INTRODUCTION

During the Covid-19 pandemic, almost all jobs and activities to be limited, relying on the internet network. Data security in an internet network is most important and needs to be a concern for internet users.

The internet network is a public network, which is vulnerable to security attacks. Attacks may occur to retrieve user data in the form of a username and password

OTP Verification is the process of verifying a user by sending a unique password so that the user can be verified before completing a registration or payment process.

Most of the time, we get an OTP when we make an online payment, or when we forget our password, or when creating an account on any online platform.

Thus, the sole purpose of an OTP is to verify the identity of a user by sending a unique password.

OTP is classified into two types : HOTP and TOTP

HMAC-based One-time Password algorithm (HOTP) is an event-based OTP where the moving factor in each code is based on counter.

Time-based One-time Password (TOTP) is a time-based OTP where the moving factor in each code is based on time.

Every HOTP code is valid until it's used, or until a subsequent one is validated by the server. So the chances for hacking are high.

TOTP is basically a branch of HOTP and it has a huge advantage over HOTP.

Unlike with HOTP, OTPs are generated using the number of time steps from Unix time. Usually, the time step is set to either 30 or 60 secs.

So when considering TOTP vs HOTP the obvious choice is TOTP, simply because it is more secure.

2. LITERATURE SURVEY

2.1. Title :

TOTP : Time-Based One-Time Password Algorithm

2.2. Author(s) :

David M'Raihi , Salah Machani , Mingliang Pei , Johan Rydell .

2.3. Reference link :

RFC 6238 - TOTP: Time-Based One-Time Password Algorithm (ietf.org)

2.4. Critical Assessment :

2.4.1. Previous study :

The HOTP algorithm is based on an increasing counter value and a static symmetric key known only to the token and the validation service. In order to create the HOTP value, we will use the HMAC -SHA -1 algorithm.

As the output of the HMAC-SHA-1 calculation is 160 bits, we must truncate this value to something that can be easily entered by a user.

HOTP(K,C) = Truncate(HMAC – SHA – 1(K,C))

Where :

- Truncate represents the function that converts an HMAC-SHA-1 value into an HOTP value.

The Key (K), the Counter (c), and Data values are hashed high-order byte first.

2.4.2. Current study :

Basically, we define TOTP as *TOTP = HOTP(K, T)*, where *T* is an integer and represents the number of time steps between the initial counter time *T0* and the current Unix time.

More specifically, *T = (Current Unix time – T0) / X*, where the default floor function is used in the computation.

For example, with *T0 = 0* and Time Step *X = 30*, *T = 1* if the current Unix time is 59 seconds, and *T = 2* if the current Unix time is 60 seconds.

The implementation of this algorithm must support a time value *T* larger than 32-bit integer when it is beyond the year 2038. The value of the system parameters *X* and *T0* are pre- established during the provisioning process and communicated between a prover and verifier as part of the provisioning step.

3. HARDWARE AND SOFTWARE REQUIREMENTS

3.1. Hardware Requirements :

Device name	**:** DESKTOP-9U38MCH
Processor	**:** Intel® Core™ i7-8565U CPU @1.80GHz 1.99 GHz
Installed RAM	**:** 16.0 GB (15.8 GB usable)
Device ID	**:** 2F0DA166-C34F-44D9-80DC-1264CC873D6C
Product ID	**:** 00331-10000-00001-AA182
System type	**:** 64-bit operating system, x64-based processor

3.2. Software Requirements :

Language	: Python 3.6 or +
Framework	: Django 3.0 or +
Tools	: Pycharm IDE, Web browser

4. FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

4.1. Functional requirements :

- The required python packages should be installed :
  django, pillow, qrcode, pyotp, time, etc.
- Authentication is mandatory.
- QR code must be displayed.
- OTP must be generated after scanning the QR code.

4.2. Non-Functional requirements :

- Text box and buttons should be displayed in proper alignment.
- The cursor should be working.
- Page rendering should work properly.

5. PROPOSED SYSTEM

There are different algorithms for OTP generation. The most popular one is HOTP i.e. HMAC-based One-time Password algorithm. Most of the other algorithms such as TOTP (Time based One-time Password) algorithm and OCRA (OATH challenge-response) algorithm are based on HOTP.

In our work, we propose to analyze these algorithms and implement HOTP and TOTP verification systems.

5.1. Methods :

5.1.1. HOTP :

- HOTP stands for Hash-Based OTP Verification.
- HOTP is HMAC based OTP algorithm, also referred to as event-based one-time Pass.
- HMAC Algorithm stands for Hash-Based Message Authentication Code.
- HOTP is a Counter Based OTP Verification.
- This algorithm takes some time to deliver a message with a one-time OTP and can be valid for a longer period.
- HOTP is least secured compared to other OTP generation algorithms.
- PROBLEMS:
    - ➢ HOTP has Synchronization Problem.
    - ➢ HOTP has Security Problem.

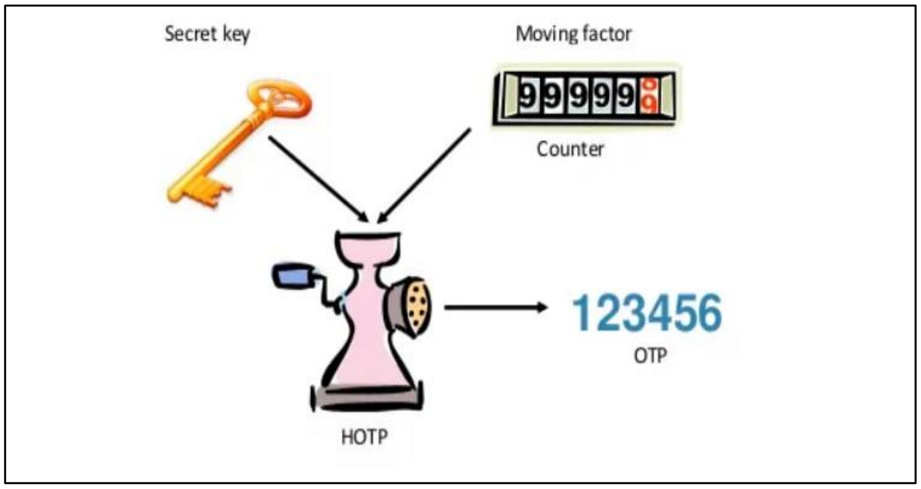


Fig : 1.1

5.1.2. TOTP :

- TOTP stands for Time-Based OTP Verification.
- TOTP is Token Based where the TOTP token has a secret key and the current time value.
- TOTP will take 30 sec to deliver a code or OTP and that OTP will be valid only for 30 sec, and after 30 sec the code will get expired.
- TOTP is more secure compared to HOTP.
- There are two parameters used to generate OTP using the TOTP Algorithm
  - ➢ The Shared Secret (Unique code, generally 16-32 Base 32 character long).
  - ➢ The current time interval (usually 30 or 60 sec).

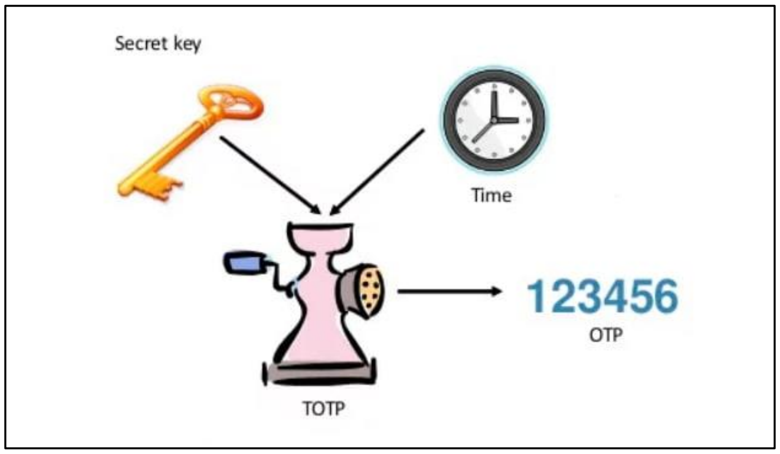


Fig : 1.2

5.1.3. OCRA:

- OCRA or OATH challenge-response algorithm is the most reliable multi-factor authentication algorithm.
- OCRA algorithm is proved to be the safest one created by the OATH (Open Authentication initiative) as it includes one-time passcode generation alongside the secret key and a counter or time.
- The challenge-response algorithm (OCRA) can be identified as advanced HOTP.
- OCRA algorithm expanded TOTP further by introducing the challenge-response mode to calculate OTP values

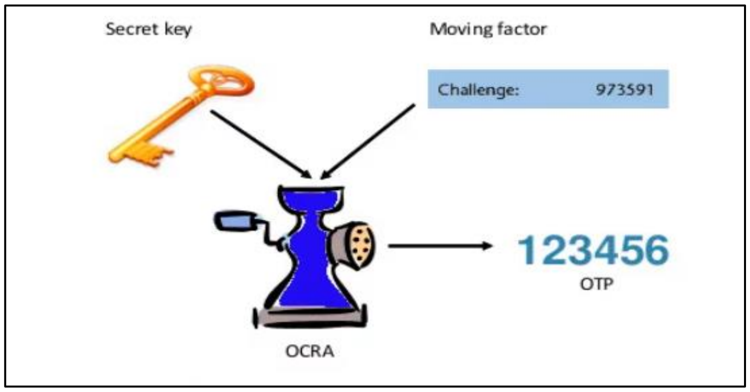
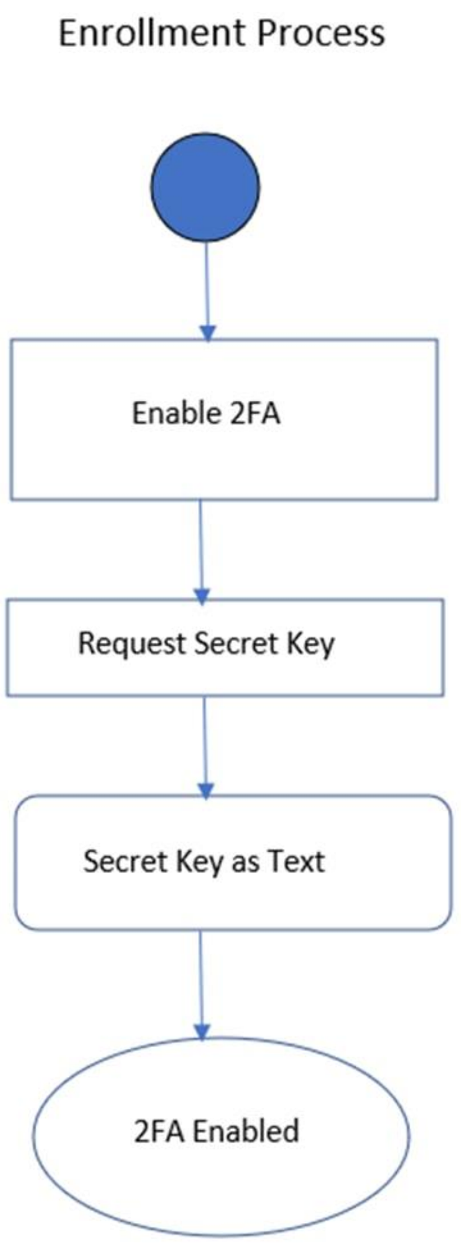


Fig : 1.3

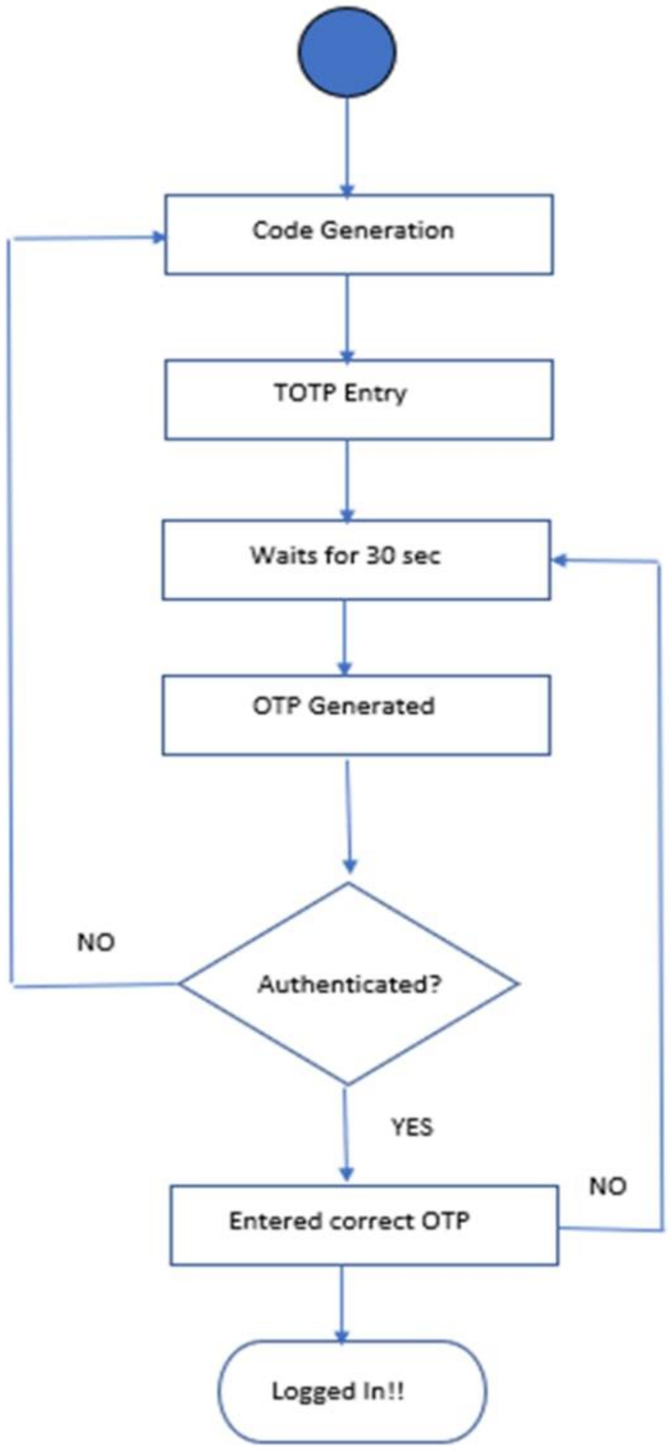
6. IMPLEMENTATION

6.1. Console :

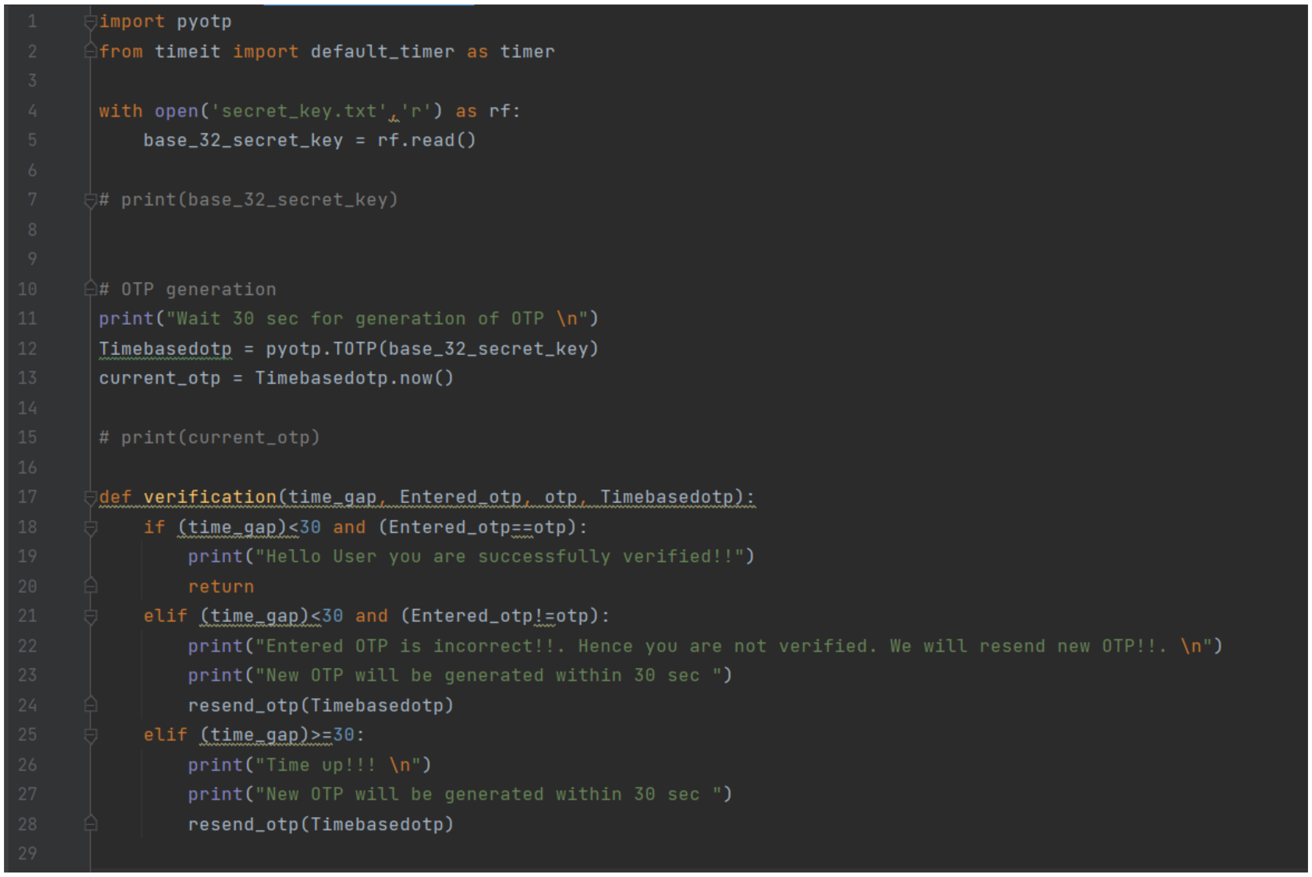
6.1.1. Flowchart :



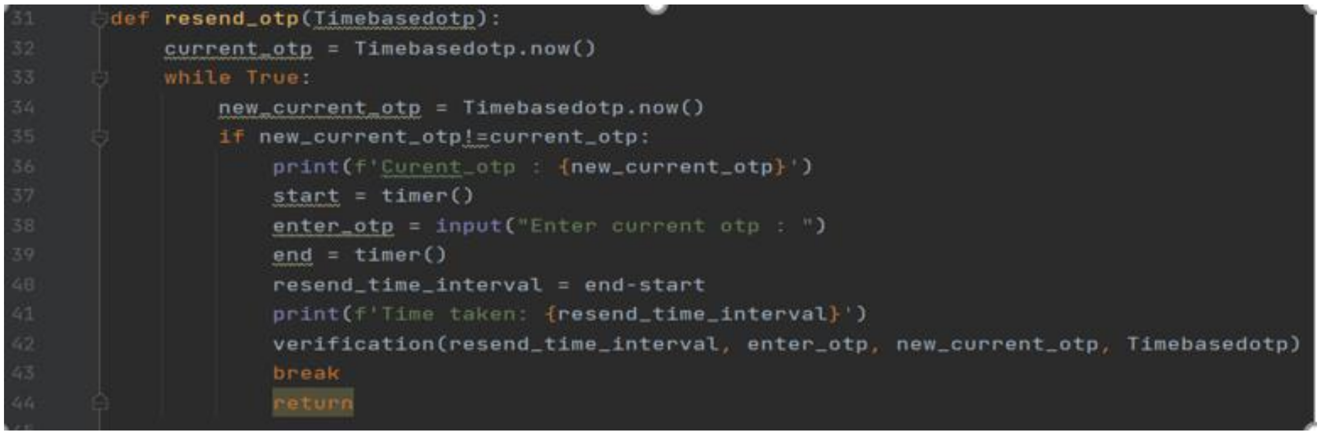
Login Process



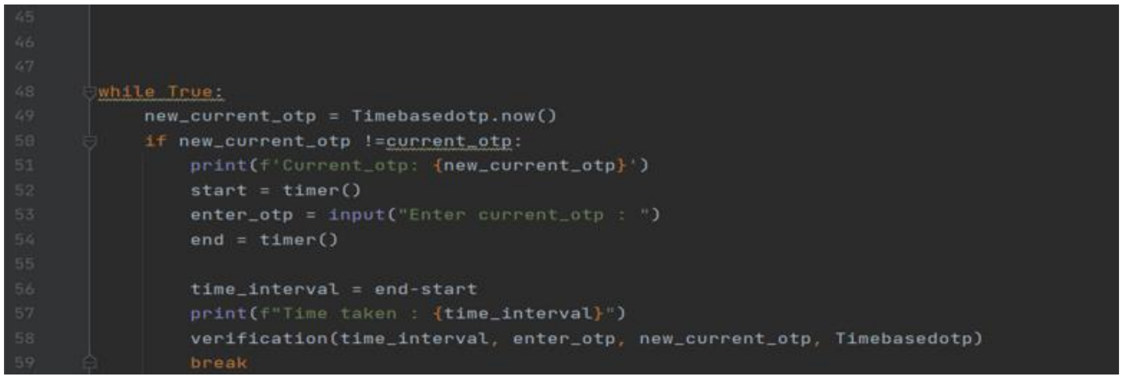
6.1.2. Code :



```python
import pyotp
from timeit import default_timer as timer

with open('secret_key.txt','r') as rf:
    base_32_secret_key = rf.read()

# print(base_32_secret_key)


# OTP generation
print("Wait 30 sec for generation of OTP \n")
Timebasedotp = pyotp.TOTP(base_32_secret_key)
current_otp = Timebasedotp.now()

# print(current_otp)

def verification(time_gap, Entered_otp, otp, Timebasedotp):
    if (time_gap)<30 and (Entered_otp==otp):
        print("Hello User you are successfully verified!!")
        return
    elif (time_gap)<30 and (Entered_otp!=otp):
        print("Entered OTP is incorrect!!. Hence you are not verified. We will resend new OTP!!. \n")
        print("New OTP will be generated within 30 sec ")
        resend_otp(Timebasedotp)
    elif (time_gap)>=30:
        print("Time up!!! \n")
        print("New OTP will be generated within 30 sec ")
        resend_otp(Timebasedotp)

```



```python
def resend_otp(Timebasedotp):
    current_otp = Timebasedotp.now()
    while True:
        new_current_otp = Timebasedotp.now()
        if new_current_otp!=current_otp:
            print(f'Curent_otp : {new_current_otp}')
            start = timer()
            enter_otp = input("Enter current otp : ")
            end = timer()
            resend_time_interval = end-start
            print(f'Time taken: {resend_time_interval}')
            verification(resend_time_interval, enter_otp, new_current_otp, Timebasedotp)
            break
            return
```

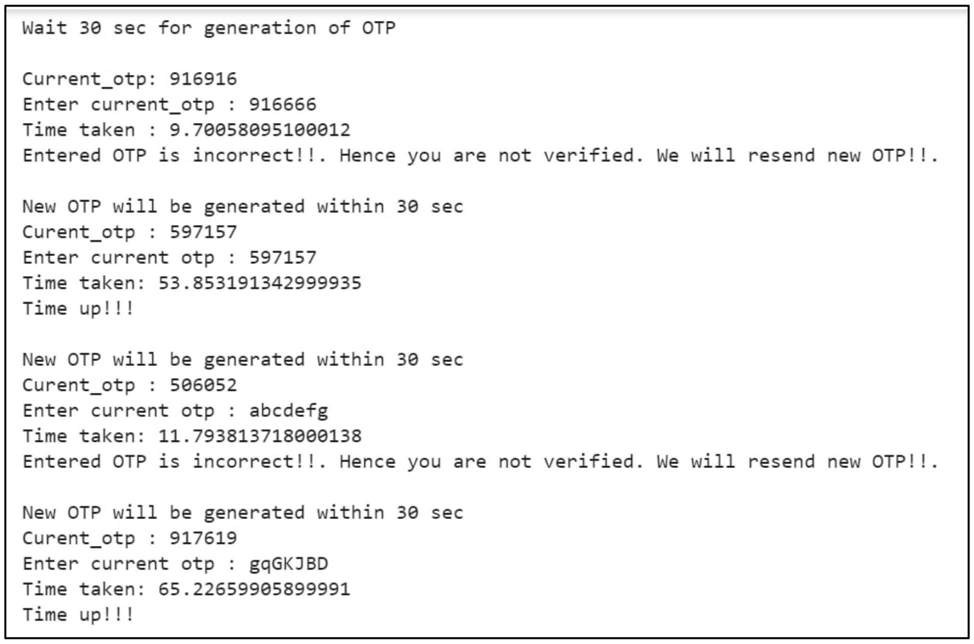


```
while True:
    new_current_otp = Timebasedotp.now()
    if new_current_otp !=current_otp:
        print(f'Current_otp: {new_current_otp}')
        start = timer()
        enter_otp = input("Enter current_otp : ")
        end = timer()

        time_interval = end-start
        print(f"Time taken : {time_interval}")
        verification(time_interval, enter_otp, new_current_otp, Timebasedotp)
        break
```

6.1.3. Code Description :

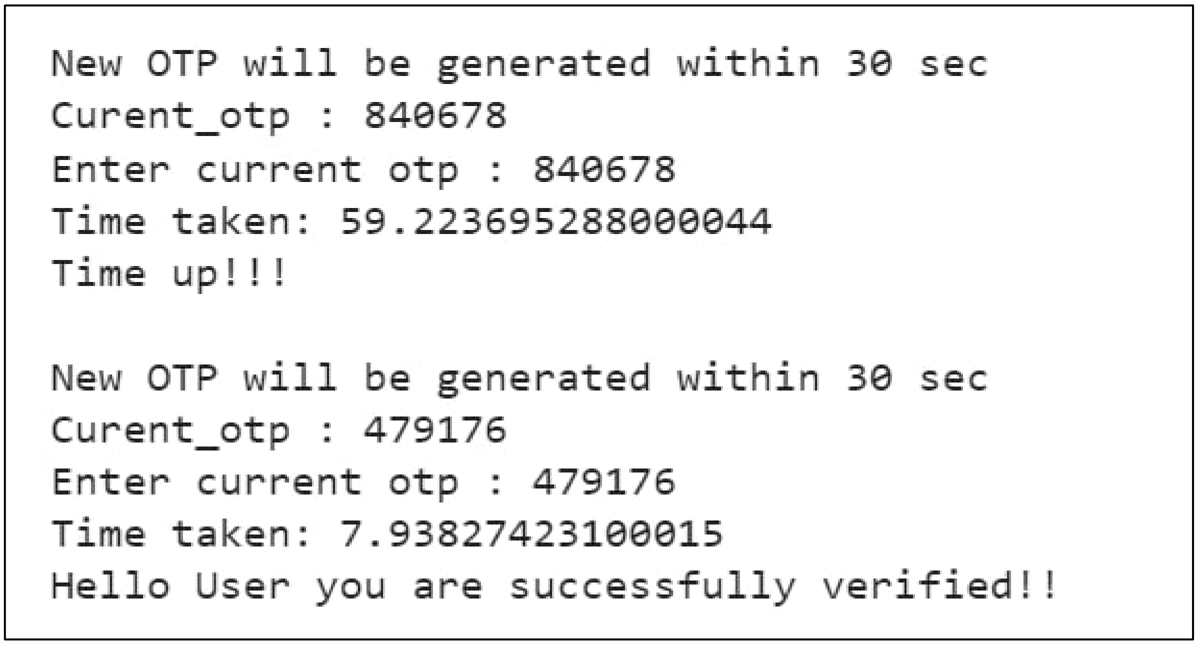
- First we need to import “pyotp” library. Next the “secret_key.txt” file is opened and the secret code is read with the help of “read()” method.
- The “Timebasedotp” variable stores the secret code value and it is assigned to “current_otp” variable.
- The “verification()” method is used whether the entered OTP is correct or not. If the entered OTP is correct within less than 30 seconds then the user can login successfully. If the entered OTP is not correct within less than 30 seconds then another OTP is resent. If the user doesn’t enter the OTP within 30 seconds then another OTP is resent.
- To resend the OTP, “resend()” method is used. This method sends the new OTP code and also calculates the time taken by the user to enter the OTP.

6.1.4. Outputs :



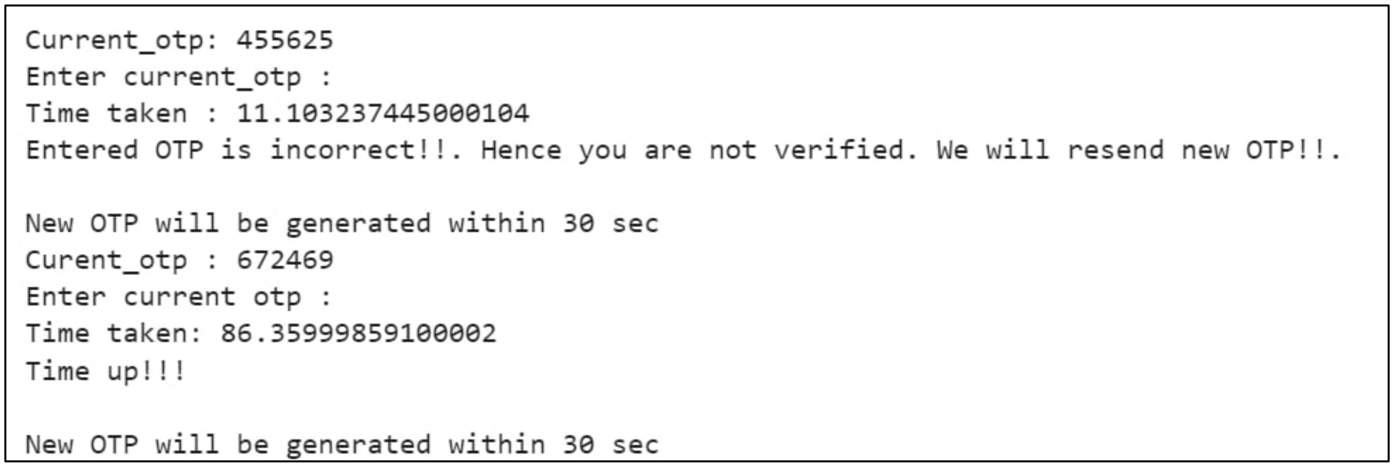
```
Wait 30 sec for generation of OTP

Current_otp: 916916
Enter current_otp : 916666
Time taken : 9.70058095100012
Entered OTP is incorrect!!. Hence you are not verified. We will resend new OTP!!.

New OTP will be generated within 30 sec
Curent_otp : 597157
Enter current otp : 597157
Time taken: 53.853191342999935
Time up!!!

New OTP will be generated within 30 sec
Curent_otp : 506052
Enter current otp : abcdefg
Time taken: 11.793813718000138
Entered OTP is incorrect!!. Hence you are not verified. We will resend new OTP!!.

New OTP will be generated within 30 sec
Curent_otp : 917619
Enter current otp : gqGKJBD
Time taken: 65.22659905899991
Time up!!!
```

Fig : 2.1



```
New OTP will be generated within 30 sec
Curent_otp : 840678
Enter current otp : 840678
Time taken: 59.223695288000044
Time up!!!

New OTP will be generated within 30 sec
Curent_otp : 479176
Enter current otp : 479176
Time taken: 7.93827423100015
Hello User you are successfully verified!!
```

Fig : 2.2

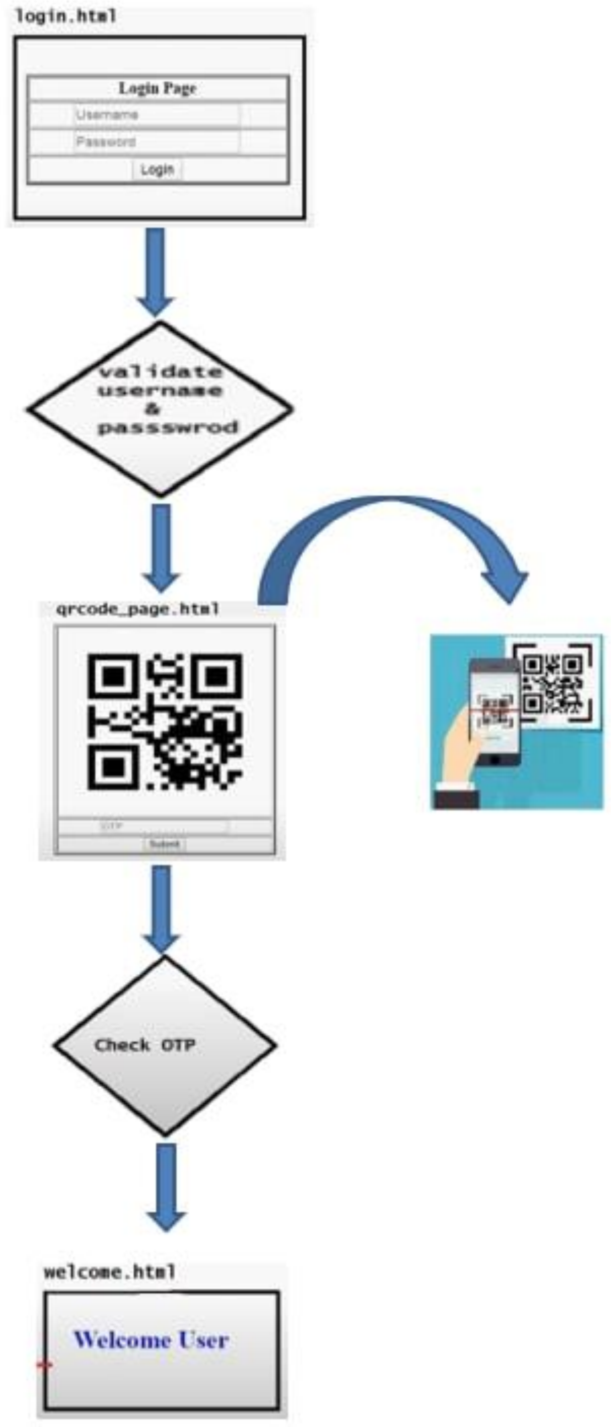


```
Current_otp: 455625
Enter current_otp :
Time taken : 11.103237445000104
Entered OTP is incorrect!!. Hence you are not verified. We will resend new OTP!!.

New OTP will be generated within 30 sec
Curent_otp : 672469
Enter current otp :
Time taken: 86.35999859100002
Time up!!!

New OTP will be generated within 30 sec
```

Fig : 2.3

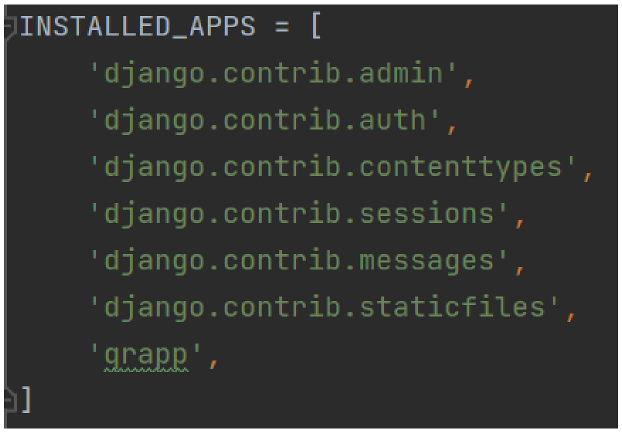
6.2. Front-end :

6.2.1. Flowchart :

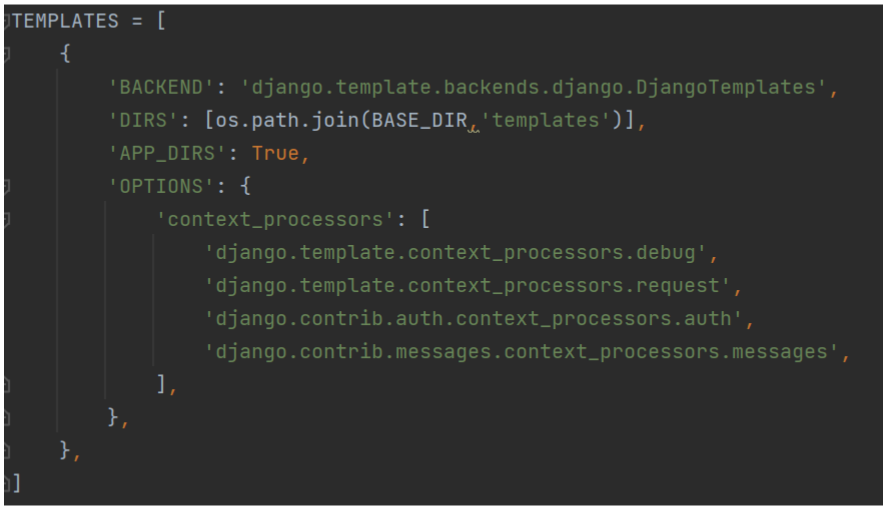


6.2.2. Code :

settings.py



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'qrapp',
]
```



```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

urls.py

```
from django.contrib import admin
from django.urls import path
from qrapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.openLoginPage),
    path('validate',views.validateuser,name="validate_login"),
    path('validate_otp',views.validateOTP,name="validate_otp"),
]
```

views.py

```
from django.shortcuts import render
import random
import qrcode  #pip install qrcode and also pip install pillow (library)

#to check
otp = 0 #global value
```

```
#login page
def openLoginPage(request):
    return render(request,"login.html")

#to check validate username and password
def validateuser(request):
    username = request.POST.get("t1")
    password = request.POST.get("t2")

    if username == "harika" and password == "1234": #create QR
      #random number is generated
      rno = random.randint(100000,999999)
      global otp
      otp = rno
      im = qrcode.make("OTP : "+str(rno)) #write the rno in the image
      im.save(r"qrapp/static/qrimages/image.jpg")   #save the image in the app(qrapp)
      return render(request,"qrcode_page.html")
    else:
        return render(request,"login.html",{"message":"Invalid User"})

#checking if the entered otp is correct or wrong
def validateOTP(request):
        user_otp = request.POST.get("otp")
        if user_otp == str(otp): #open welcome
            return render(request,"welcome.html")
        else:
            return render(request, "login.html", {"message": "Invalid OTP"})
```

login.html

```
<html>
<body>
<form action="{% url 'validate_login' %}" method="post">
    {% csrf_token %}
    <h1 align="center">ONE TIME PASSWORD VERIFICATION SYSTEM</h1>

    <table align="center" border="2">
    <tr>
        <th width="250">Login Page</th>
    </tr>
    <tr>
        <th>
            <input type="text" placeholder="Username" name="t1" required>
        </th>
    </tr>
    <tr>
        <th>
            <input type="password" placeholder="Password" name="t2" required>
        </th>
    </tr>
    <tr>
        <th>
            <button type="submit">Login</button>
        </th>
    </tr>
</table>
</form>
```

```
{% if message %}
<h1 style="color: red">{{message}}</h1>
{% endif %}
</body>
</html>
```

qrcode_page.html

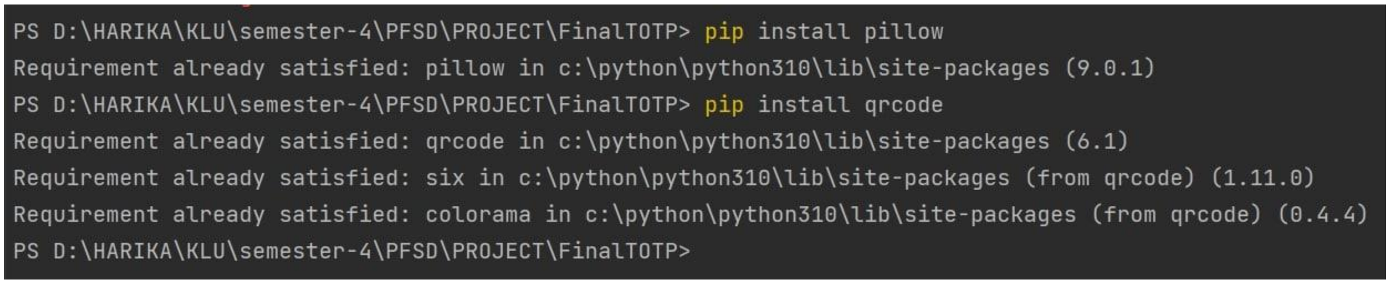
```
{% load static %}
<html>
<body>
<form action="{% url 'validate_otp' %}" method="post">
{% csrf_token %}
   <h1 align="center">SCAN THE BELOW QR CODE TO GET OTP</h1>
<table align="center" border="2">
<tr>
  <th>
    <img src="{% static 'qrimages/image.jpg' %}">
  </th>
</tr>
<tr>
  <th>
    <input type="password" placeholder="OTP" required name="otp">
  </th>
</tr>
<tr>
  <th>
    <button type="submit">Check</button>
  </th>
</tr>
</table>
</form>
</body>
</html>
```

welcome.html

```
<html>
<body>
<h1 style="color: black" align="center">WELCOME... YOU HAVE LOGGED-IN
SUCCESSFULLY</h1>
<h3>Team Members</h3>
<h3>TEJASWI KOTTAKKI - 2010030088</h3>
<h3>HARIKA NETHI - 2010030119</h3>
<h3>SHREYA CHINTAWAR - 2010030155</h3>
<h3>AMRUTHA VARSHINI - 2010030009</h3>
</body>
</html>
```

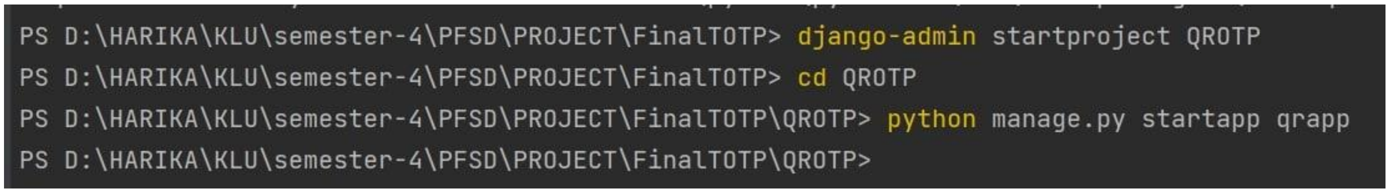
6.2.3. Code Description :

- Firstly, we need to install pillow and qrcode packages. For this we use the below commands



```
PS D:\HARIKA\KLU\semester-4\PFSD\PROJECT\FinalTOTP> pip install pillow
Requirement already satisfied: pillow in c:\python\python310\lib\site-packages (9.0.1)
PS D:\HARIKA\KLU\semester-4\PFSD\PROJECT\FinalTOTP> pip install qrcode
Requirement already satisfied: qrcode in c:\python\python310\lib\site-packages (6.1)
Requirement already satisfied: six in c:\python\python310\lib\site-packages (from qrcode) (1.11.0)
Requirement already satisfied: colorama in c:\python\python310\lib\site-packages (from qrcode) (0.4.4)
PS D:\HARIKA\KLU\semester-4\PFSD\PROJECT\FinalTOTP>
```

- We need to create Django project (QROTP) and Django app (qrapp)



```
PS D:\HARIKA\KLU\semester-4\PFSD\PROJECT\FinalTOTP> django-admin startproject QROTP
PS D:\HARIKA\KLU\semester-4\PFSD\PROJECT\FinalTOTP> cd QROTP
PS D:\HARIKA\KLU\semester-4\PFSD\PROJECT\FinalTOTP\QROTP> python manage.py startapp qrapp
PS D:\HARIKA\KLU\semester-4\PFSD\PROJECT\FinalTOTP\QROTP>
```

- views.py file
  - ➢ openloginpage() :

    This method is used to display the login page after executing the code.
  - ➢ validateuser() :

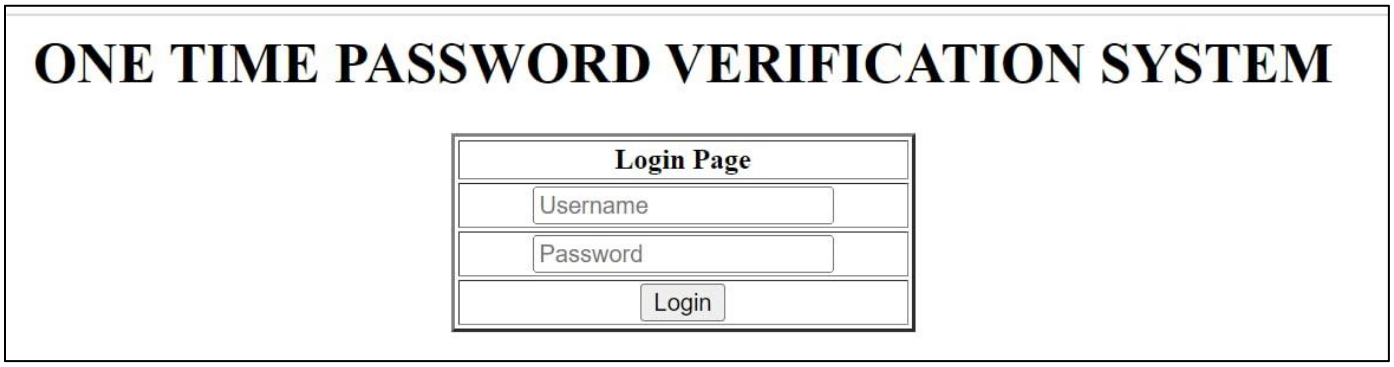
    This method is used to check whether the user has given the correct username and password. If the username and password are correct, a qrcode is generated with random number and it will render to the qrpage and displays a QR Code to scan.

    If the username and password are incorrect it will render to the login page with a message “Invalid User” in red color.
  - ➢ validateOTP() :

    This method is used to check whether the user has entered a valid OTP or not.

    If the OTP is correct, then it’ll render to the welcome page. Else it redirects to the login page with a message “Invalid OTP” in red color.

6.2.4. Outputs :

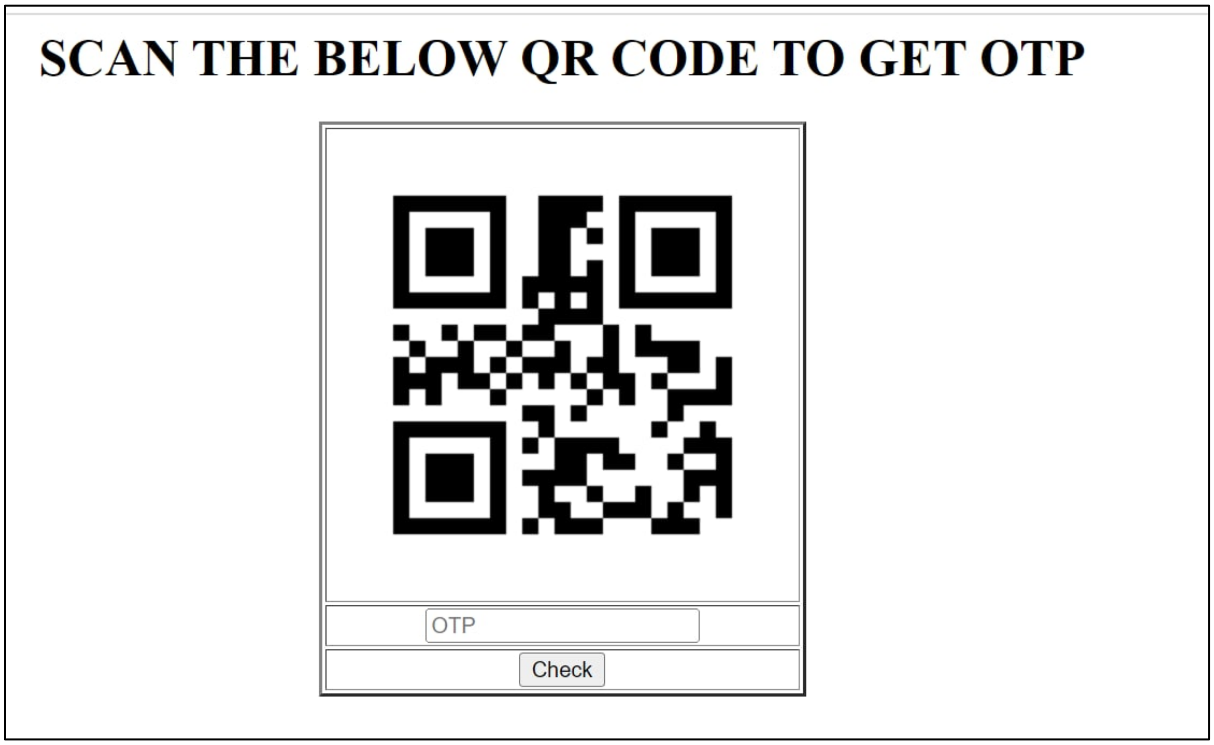

ONE TIME PASSWORD VERIFICATION SYSTEM

Login Page

Username

Password

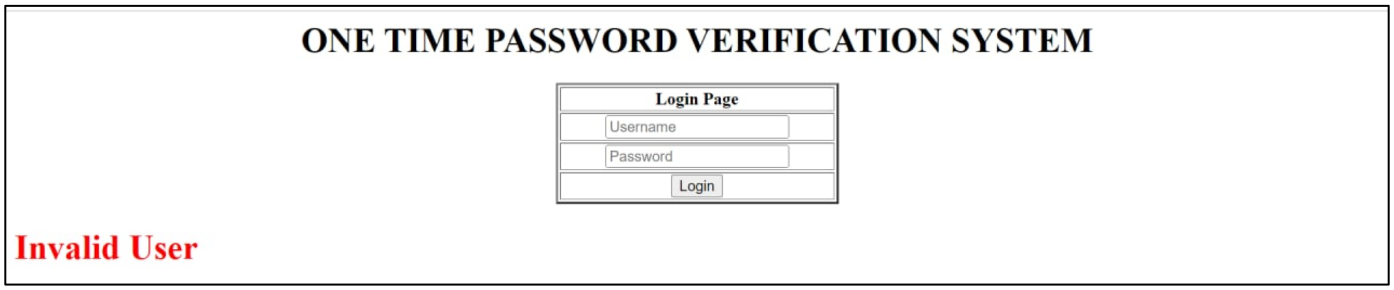
Fig : 3.1


SCAN THE BELOW QR CODE TO GET OTP

OTP

Fig : 3.2

If the given credentials in the login page are valid then the above QR code will be displayed


ONE TIME PASSWORD VERIFICATION SYSTEM

Login Page

Invalid User

Fig : 3.3

If the given credentials in the login page are invalid then it will render to the login page and the above message will be displayed

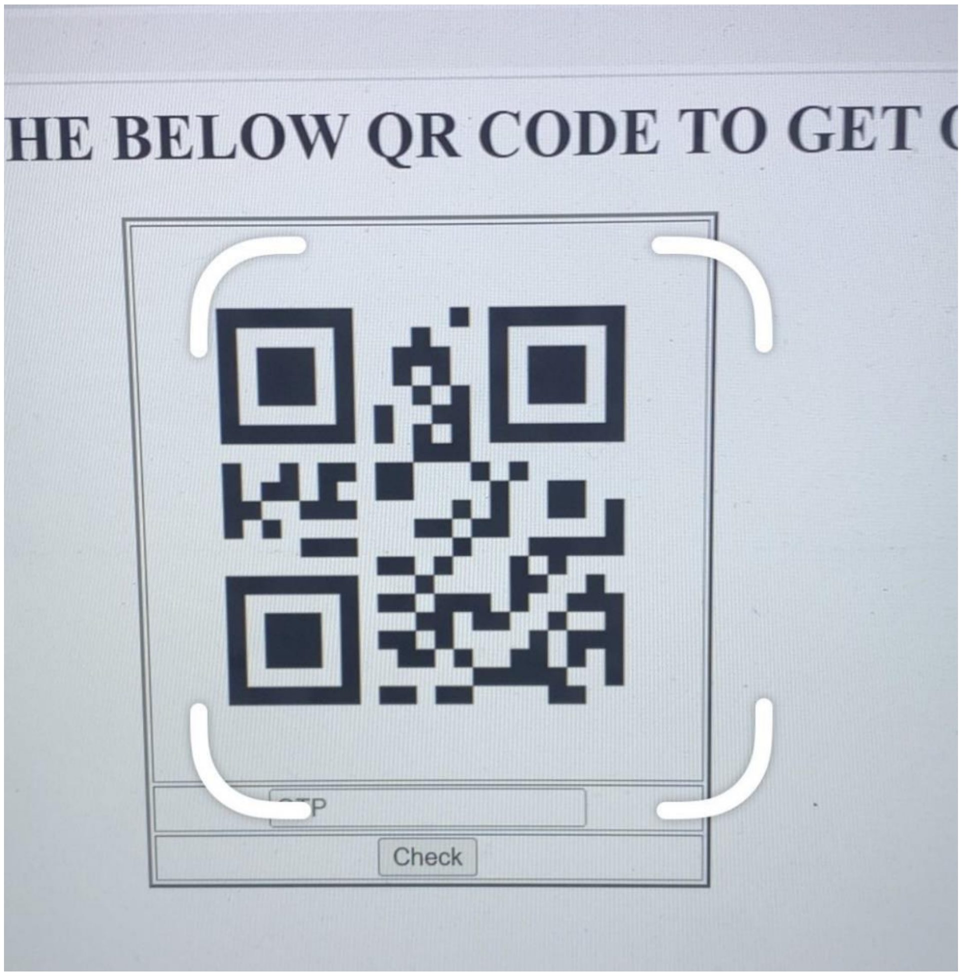
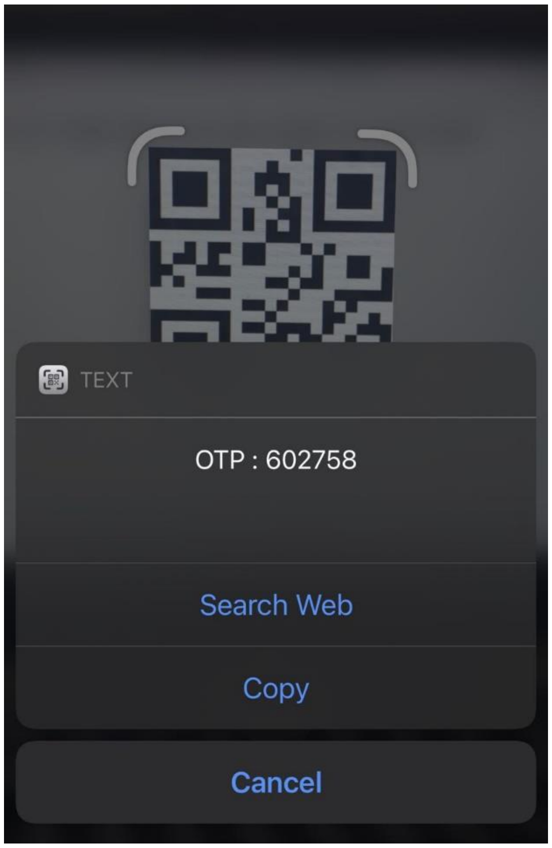

HE BELOW QR CODE TO GET

Fig : 3.4

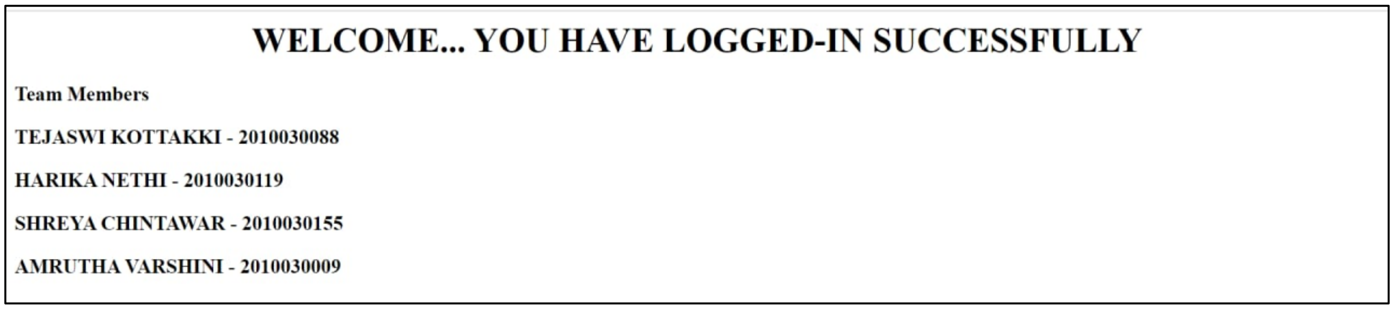
Scanning the QR code



OTP : 602758

Fig : 3.5

The OTP is generated



WELCOME... YOU HAVE LOGGED-IN SUCCESSFULLY

Team Members

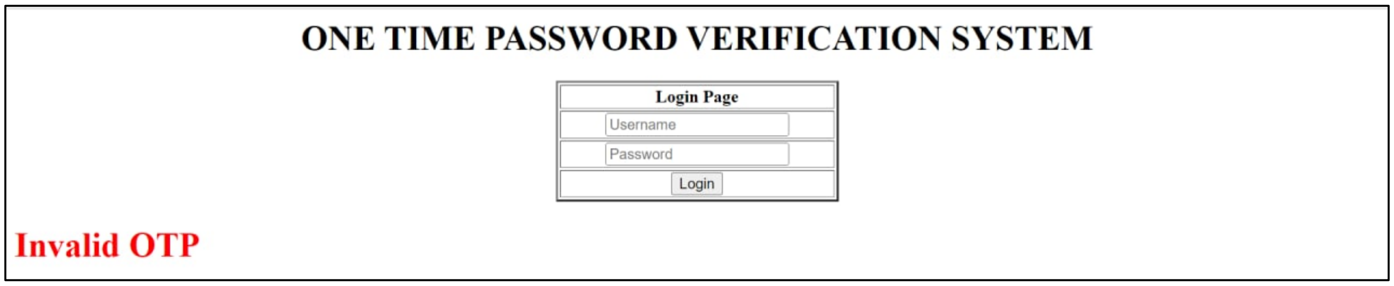
TEJASWI KOTTAKKI - 2010030088

HARIKA NETHI - 2010030119

SHREYA CHINTAWAR - 2010030155

AMRUTHA VARSHINI - 2010030009

Fig : 3.6
If the given OTP is correct it will render to the welcome page


ONE TIME PASSWORD VERIFICATION SYSTEM

Invalid OTP

Fig : 3.7

If the entered OTP is invalid then it will render to the login page and the above message will be displayed.

7. RESULTS AND DISCUSSION

7.1. Results :

Time-based One Time Password or TOTP for short, does not utilize a counter for the server-user synchronization but generates a password based on the current time. The advantage of the TOTP password is a limited lifetime, usually 30-60 seconds.
TOTP algorithm is a branch of HOTP, TOTP is based on time and HOTP is based on Counter value. The security and strength of the TOTP algorithm depend on the properties of the underlying building block HOTP, which is a construction based on HMAC using SHA-1 as the hash function.
One of the greatest advantages of TOTP is that it does not need any Internet connection for the OTP to get verified and it is easy to use across applications and channels.

7.2. Discussion :

The conclusion of the security analysis is that, for all practical purposes, the outputs of the dynamic truncation on distinct inputs are uniformly and independently distributed strings.
The analysis demonstrates that the best possible attack against the HOTP function is the brute force attack.
As indicated in the algorithm requirement section, keys should be chosen at random or using a cryptographically strong pseudorandom generator properly seeded with a random value.
Keys should be of the length of the HMAC output to facilitate interoperability.
All the communications should take place over a secure channel, e.g., Secure Socket Layer/Transport Layer Security (SSL/TLS) or IPsec connections.
It is recommended to store the keys securely in the validation system, and, more specifically, encrypt them using tamper-resistant hardware encryption and exposing them only when required: for example, the key is decrypted when needed to verify an OTP value and re-encrypted immediately to limit exposure in the RAM to a short period of time.
The key store must be in a secure area, to avoid, as much as possible, a direct attack on the validation system and secrets database. Particularly, access to the key material should be limited to programs and processes required by the validation system only.
The time-step size has an impact on both security and usability. A larger time-step size means a larger validity window for an OTP to be accepted by a validation system.
A larger time-step size exposes a large window to attack. When an OTP is generated and exposed to a third party before it is consumed, the third party can consume the OTP within the time-step window.
Therefore, it is recommended to use a default time step size of 30 secs. This default value of 30 secs is selected between security and usability.

8. CONCLUSION AND FUTURE WORK

8.1. Conclusion :

TOTP, or Time-based OTP, is basically a branch of HOTP. And it has a huge advantage over HOTP — instead of the HOTP counter, TOTP tokens use time (UNIX time plus time-steps). Like with HOTP the user and server share a seed on setup. Unlike with HOTP — after that, the OTPs are generated using the number of time steps from the UNIX time. Usually, the time step is set to either 30 or 60 secs.

OATH HOTP-compatible tokens generate OTPs that do not have an expiration period. And we have already come to the conclusion that this creates a major security vulnerability. TOTP passcodes, on the other hand, have the advantage of being valid for a limited time period — the time step. So if the generated pass is not used within the 30-60 seconds it expires and cannot be used for login.

So when considering TOTP vs HOTP the obvious choice is TOTP, simply because it is more secure.

8.2. Future Work :

- We can further extend our project to mini app.
- We can integrate our project with other projects to increase security services.

9. BIBLIOGRAPHY

https://thecleverprogrammer.com/2021/04/14/otp-verification-using-python/

https://iopscience.iop.org/article/10.1088/1742-6596/1783/1/012041/pdf

http://article.nadiapub.com/IJGDC/vol10_no11/5.pdf

https://sci-hub.se/https://link.springer.com/chapter/10.1007/978-3-319-58808-7_5

https://sci-hub.se/https://ieeexplore.ieee.org/abstract/document/7375661

http://www.researchinventy.com/papers/v2i10/C0210014017.pdf

https://www.hjp.at/doc/rfc/rfc6238.html

https://datatracker.ietf.org/doc/html/rfc6238

https://nbviewer.org/github/algorithmic-space/cryptoy/blob/master/rfc6238.ipynb