

## TABLE OF CONTENTS:

1.ABSTRACT.....	1
2.INTRODUCTION... ..	2
3.OBJECTIVES OF THE PROJECT... ..	5
4.LITERATURE SURVEY.....	7
5.EXISTING SYSTEM... ..	8
5.1. DISADVANTAGES.....	8
6.PROPOSED SYSTEM... ..	9
6.1. ADVANTAGES... ..	9
7.ARCHITECTURE AND DESIGN.....	10
8.SYSTEM AND SOFTWARE REQUIREMENTS... ..	12
9.UML DIAGRAMS... ..	13
10.METHOD/IMPLEMENTATION/ALGORITHM .....	17
11.IMPLEMENTATION.....	19
11.1. CODE.....	19
11.2. RESULTS.....	21
11.3. SCREENSHOT WITH DISCUSSION.....	22
12.CONCLUSION.....	23
13.REFERENCES... ..	24

## **ABSTRACT**

Static password is no more considered as a secure alternative to attain the authentication of the valid user in the internet world. This already opens the door for the attackers to compromise the password because of several vulnerable security attacks. Thus, it needs a secure mechanism which introduces the concept of OTP where password is frequently changed and valid for a certain period. The concept makes the OTP a strong alternative of static password that is widely used now a day. However, now smart phone is convenient and commonly used media to receive OTP generated by the system which is threatened by the man-in-the-middle attack if the channel for transferring the OTP to the user end is not protected. Bearing this shortcoming of existing OTP system, we develop a method where the user will perform a certain intermediate mathematical calculation before using the OTP received by the smart phone and thus, incorporates an extra layer of security in the existing OTP system. This extra layer of security protects the method from usual security attacks whether the channel used for transferring OTP is compromised or not. This new method of OTP has noticeable improvements in performance over the existing system and puts no extra burden in the server-side verification process. The purpose of implementing an OTP verification system is to ensure that only authorized users can access sensitive information or perform important actions. By requiring users to enter a valid OTP, the system helps to prevent unauthorized access, identity theft, and fraudulent activities.

## **INTRODUCTION**

A One-time Password is a password or code that is automatically generated and sent to a digital device to allow a single login session or transaction. Also known as a One-time PIN, one-time authorization code (OTAC), One-Time-Pass Code, or dynamic password, OTP mitigates several risks of traditional static password-based authentication.

OTPs are randomly generated numeric codes to authenticate login attempts and transactions. The unique codes are difficult to guess, adding a strong layer of security for each authentication event. OTP verification plays a critical role in 2-Factor Authentication (2FA) solutions implemented by leading financial service providers and government institutions to ensure the highest level of security for customer transactions security codes have become a popular method of enabling a single login to validate a new account or confirm a legitimate transaction. OTPs consist of numbers or a string of characters automatically generated and delivered to the user's devices by SMS, Voice, Email, or Push messages. Enterprise systems need more than static passwords to protect sensitive data. Standalone passwords can result in exposed data since they cannot effectively verify if the user trying to access data is authentic or a cyber threat actor. According to Forrester, about 80 percent of security breaches are due to compromised privileged credentials. Unfortunately, traditional passwords are not enough to account for the human element that causes most breaches globally due to weak or stolen passwords. OTPs are far more secure than static passwords because they typically expire after a short period. Unlike traditional passwords that one can use for months without changing, you can use OTP codes only once. OTP verification involves secure technology that ensures that only the authorized person can access data or an account.

You can ensure login verification by sending One-Time Passcodes (OTPs) to a user's phone number. By confirming deliverability to the right user, you can prevent malicious attacks by bots and hackers. In addition, businesses that require highly secure solutions use a combination of OTPs and passwords to minimize the risk of fraudulent activities. A One Time Password is usually a six-digit number sent to a user's device via SMS message, email, or voice message. An OTP platform involves an authentication server, which verifies the information a user enters and prompts for a code. The authentication server generates and sends the OTP message to the user. Then, the authentication server verifies the OTP entered by the user and authenticates the transaction or account login.

OTP values are generated using the Hashed Message Authentication Code (HMAC) algorithm and a moving factor. The two common types of OTPs are Hash-based OTPs (HOTP) and Time-based OTPs (TOTP).

HOTP vs TOTP

## **HOTP**

Uses the counter as a moving factor The counter is incremented after code generation Codes expire after use or a new OTP request Codes cannot be used more than once Also called event-based OTPs

## **TOTP**

Uses time as moving factor OTP values have timestamps such as exact minute and second Valid only for a certain period Codes expire after use or after a set amount of time (typically within minutes) Also called Time Synchronized OTP. SMS is the most common way of delivering OTP messages due to the easy accessibility on mobile phones. However, there are other ways to send OTPs, such as voicemails, push messages, and emails.

## **SMS OTP Verification**

When a user attempts a transaction or login, an OTP is sent as an SMS message to the mobile phone on the number linked to their account. Once the user then keys in the code, the authentication server verifies the user. Since SMS OTP services does not require an internet connection and can be sent to all mobile phones, it is a popular OTP delivery method. Moreover, since mobile phones generally need an unlock code, text OTPs add an additional layer of security. Although SMS is not intended to be an instant messaging platform, mobile networks worldwide can deliver a text message to recipients in just a few seconds. SMS messages are hence a widely used method to send one-time passwords.

## **Voice OTP**

Voice OTP involves pre-recorded messages with unique codes played over a phone call. Once the user enters the code in the voicemail, the server verifies the user, completing the authentication process. Voice OTPs are especially helpful for users with sight issues. Moreover, since the password is not stored on the user's mobile phone, it can be highly secure. Voice OTP is an excellent alternative to SMS codes and can be a fallback for SMS OTP verification.

## **Push OTP**

Push OTP delivery involves sending a unique code as a push message to the user's app. Push notification does not require a mobile signal and can be sent to phones with an internet/data connection. Furthermore, since push messages can only be received by the user already logged into the app, it is a secure way to deliver OTPs.

What are the Benefits of One-Time Passwords?

### **More Secure**

OTP PINs are simple yet complex, making them practically impossible to hack. Moreover, they are highly secure since they are unpredictable and not stored on a computer. The primary benefit of OTP authentication compared to standalone passwords is that they are safe from replay attacks. For example, if a cyber threat actor gets hold of your OTP, they cannot use it again because it is valid only for a single session. For the next login attempt or transaction, a new, random OTP is generated.

### **Easy to Use**

One-time codes are standard practice for everything from activating a bank card to resetting a password. Most people own a mobile phone, and SMS is available on all devices. Because SMS is so common, one-time passwords are convenient to use.

### **Highly Reliable**

OTPs sent through reliable channels such as SMS and voice are typically delivered in minutes or less. If a user does not receive OTP on time, they can request another OTP, and the authentication server attempts to resend the OTP for verification.

### **Multiple Verification Tasks**

OTP (One-Time Passwords) are pretty common in the financial industry, but they are increasingly becoming more prevalent on several websites and applications to authenticate legitimate access to data. OTPs can be used to reset forgotten passwords, complete a transaction, sign up or log into accounts, and verify online purchases. OTP also helps reduce friction in the customer journey. For example, lost/forgotten passwords can lead to drop-offs, and OTPs can help users quickly regain access to their accounts. Moreover, SMS and Voice OTP can help user's complete authentication on their mobile devices, avoiding the risks of using public computers with unsecured Wi-Fi connections.

## **OBJECTIVE OF THE PROJECT**

OTP exists to prevent unauthorized access. Companies should ensure that unauthorized access is not allowed and also authorized users cannot make unnecessary modifications. The controls exist in a variety of forms, from Identification Badges and passwords to access authentication protocols and security measures.

### **1. Enhancing Security**

One of the primary objectives is to strengthen the security of user accounts and transactions. By implementing an OTP verification system, you add an extra layer of protection against unauthorized access. Each time a user logs in or performs a transaction, they will receive a unique OTP that they need to enter to verify their identity. This helps prevent unauthorized access and protects sensitive information.

### **2. Preventing Fraud**

Another important objective is to prevent fraudulent activities, such as identity theft and unauthorized transactions. With an OTP verification system, only users with access to the registered mobile number or email address can receive the OTP. This ensures that only legitimate users can perform actions or access sensitive data, reducing the risk of fraud.

### **3. User Convenience**

While security is crucial, it's also important to consider user convenience. The OTP verification system should be designed to provide a seamless user experience. Users should be able to receive the OTP quickly and easily, without any unnecessary delays or complications. The process of entering the OTP should be straightforward and user-friendly, ensuring a positive user experience.

### **4. Scalability and Integration**

The system should be scalable to handle a large number of users and easily integrate into existing applications or platforms. As your user base grows, the system should be able to handle the increased load without compromising performance.

## **5. Error Handling and Recovery**

An important objective is to handle errors gracefully and provide appropriate recovery mechanisms. Sometimes, OTPs may fail to deliver or users may enter them incorrectly. The system should have mechanisms in place to handle these scenarios and guide users through the verification process. Clear error messages and instructions should be provided to help users resolve any issues they encounter.

## **6. Logging and Audit Trail**

Implementing proper logging and audit trail mechanisms is crucial for security and accountability. The project should include features to log OTP generation, delivery, and verification events. This allows for traceability and investigation if any suspicious activities are detected. It helps in identifying potential security breaches and ensuring accountability.

## **LITERATURE SURVEY**

Though not many systems exist in this particular field but a chrome extension to look on text messages exists and is available on the chrome store. Already existing research in the field of OTPs and mobile PC interaction are there but scarce. A study of few of them brings important and classic ideas in to the perspective. We learned about OTP generation and encryption and the comparison of 7 different algorithms for OTP generation including the S/Key and hotp: hmac based OTP algorithm etc. helped as get an overview of the scenario. We also looked on how a large number of notifications affect the system and we assess the impact of notifications on the system and it recommends that only relevant notifications are shown on the desktop. Then we studied OTP Authentication methods and the proposed Challenge Response Authentication Method for tackling the MITM or man in the middle attack for a much safer OTP transaction. We also had a brief overview of how fruitful the integration of mobile and PC could be and several limitations and problems faced were discussed. The importance of the end user was highly advocated by Kuo-Ying Huang

We read about an SMS Threat Model and the various Mobile Trojans like ZITMO Trojan.

An in-depth systematic analysis of the security of SMS authentication usage in 22 modern mobile platforms was learned by us. Various concerns were raised by researchers over SMS authentication and some ways to prevent malicious activities were presented in research paper.

We also came to know about the importance of two factor authentication or 2FA and how they help in achieving secure transactions The use of OTPs or similar texts as a means of early warning in case of disasters was researched and we found pre-emptive texts on mobiles as well as computers very necessary for the mass working continuously on computers in metropolitans. Since everything is moving to cloud so the usefulness of OTPs in cloud applications using RSA Encryption etc seems a very secure method of safeguarding cloud services.



## **EXISTING SYSTEM**

In an existing OTP verification system, users initiate authentication, prompting the system to generate a one-time password. This OTP is typically delivered via SMS or email. Upon receiving the OTP, users input it for validation, with the system verifying its correctness within a limited timeframe.

### **Disadvantages**

#### **1.Dependency on Network**

OTP verification systems typically rely on network connectivity to deliver the OTP to users. If there are network issues or delays, it can cause delays in the delivery of the OTP, leading to a frustrating user experience.

#### **2.Limited Validity Period**

OTPs usually have a limited validity period for security reasons. This means that users need to enter the OTP within a specific timeframe, which can sometimes be inconvenient if they receive the OTP when they are not immediately available to enter it.

#### **3.Inconvenience for Users**

While OTP verification adds an extra layer of security, it can also be seen as an inconvenience for users. Having to wait for an OTP, enter it correctly, and sometimes repeat the process if there are errors can be time-consuming and frustrating for users, especially if they are in a hurry.

#### **4.Dependency on Contact Information**

OTP verification systems rely on users having access to their registered mobile number or email address. If users change their contact information or lose access to their registered contact information, it can create challenges in receiving and entering the OTP.

#### **5.Potential for OTP Interception**

In rare cases, OTPs can be intercepted by malicious individuals through techniques like SIM swapping or email hacking. While these instances are relatively uncommon, they do pose a potential risk to the security of the OTP verification process.

## **PROPOSED SYSTEM**

A proposed OTP verification system would generate unique one-time passwords and deliver them securely via SMS, email, or authenticator apps. Users input the received OTP for authentication, with the system validating its correctness within a time limit. Security measures like encryption, rate limiting, and logging would ensure robust protection against unauthorized access.

### **I. Advantages**

#### **1. Biometric Verification**

One of the exciting advancements is the integration of biometric verification, such as fingerprint or facial recognition, alongside OTP verification. This adds an extra layer of security by ensuring that only the authorized user can access their account.

#### **2. Push Notifications**

Instead of relying solely on SMS or email to deliver OTPs, some systems now use push notifications directly to the user's device. This eliminates the need for users to switch between apps or check their messages, making the verification process more seamless.

#### **3. Time-based OTPs**

Traditional OTPs are typically valid for a short period of time. However, time-based OTPs generate a new code every few seconds, providing enhanced security against potential interception or replay attacks.

#### **4. Multi-Factor Authentication**

Many systems now offer multi-factor authentication, combining OTP verification with other factors like a password or security questions. This adds an extra layer of protection and makes it harder for unauthorized individuals to gain access.

#### **5. Adaptive Authentication**

Adaptive authentication systems use machine learning algorithms to analyze user behavior and determine the level of authentication required. This means that if the system detects any suspicious activity, it can prompt for additional verification steps to ensure account security.

## **ARCHITECTURE AND DESIGN**

This method is generating a 6-digit OTP and sending it to the email. The Proposed method involves OTP generation, Send OTP to respective emails, OTP validation (only 3 attempts), Send Verification Gmail. OTP is generated using python **random** library. This generated OTP is sent to Gmail for person identification. OTP generated is stored. User enters that OTP, and the system checks whether the OTP is correct or not. If the OTP entered is correct, send an OTP verified email. If the OTP entered is not correct, the system gives 3 chances to enter a correct OTP. This is Successful User Authentication.

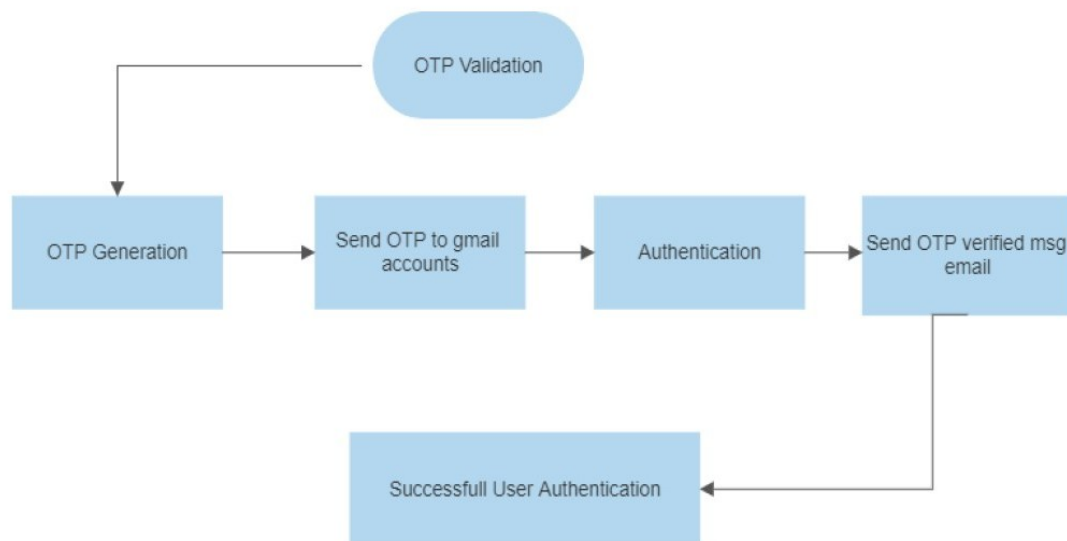


Fig1: Flow chart on OTP-Verification System

### **1. User Interface**

This is the part of the system that users interact with. It can be a mobile app, a web page, or any other interface where users enter their credentials and receive the OTP.

### **2. Application Server**

The application server is responsible for handling user requests, generating and sending the OTP, and managing the verification process. It communicates with other components to ensure a smooth user experience.

### 3. **OTP Generation**

The OTP generation component generates a unique code for each user request. It follows specific algorithms and security protocols to ensure randomness and unpredictability of the generated OTPs.

### 4. **Delivery Channels**

OTPs can be delivered through various channels like SMS, email, or push notifications. The system uses APIs or integration with third-party services to send the OTP to the user's registered contact information.

### 5. **Authentication Server**

Once the user enters the OTP, it is sent to the authentication server for verification. The server compares the entered OTP with the one generated for that specific user and determines whether it matches or not.

### 6. **User Database**

The user database stores user information, including contact details, authentication credentials, and other relevant data. It is used to validate user credentials and associate the generated OTP with the correct user.

### 7. **Security Measures**

OTP verification systems implement various security measures to protect against unauthorized access and potential attacks. This includes encryption of data, secure communication protocols, and measures to prevent brute-force attacks.

## **SYSTEM AND SOFTWARE REQUIREMENTS**

### **HARDWARE TOOLS**

### **MINIMUM REQUIREMENTS**

Processor	:	i5 or above
Hard Disk	:	1TB
RAM	:	16GB
Key Board	:	122 Keys
Mouse	:	Optical

### **SOFTWARE TOOLS**

### **MINIMUM REQUIREMENTS**

Platform	:	Windows
Operating System	:	Windows
Technology	:	Machine learning – Python
Scripting Language	:	Python
IDE	:	Jupyter notebook

## UML DIAGRAMS

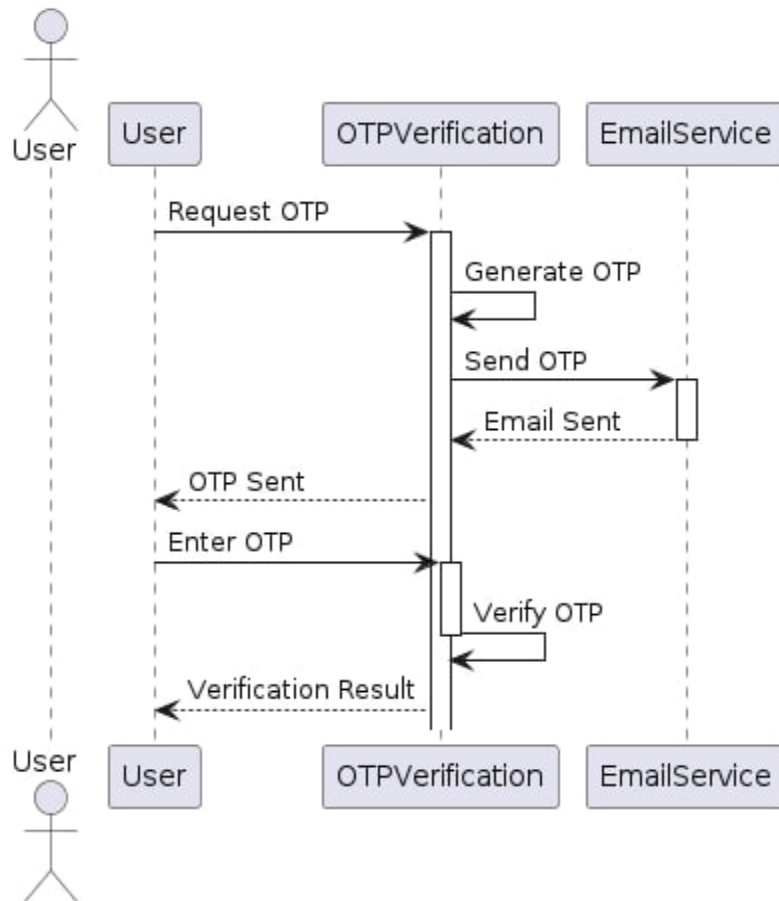


Fig 2: Sequence Diagram of OTP Verification System

A sequence diagram for OTP verification begins with the user initiating the request. The system generates and sends an OTP via the chosen communication channel. The user enters the OTP, which the system verifies. Upon successful verification, access is granted; otherwise, an error message is displayed as shown in Fig 2.

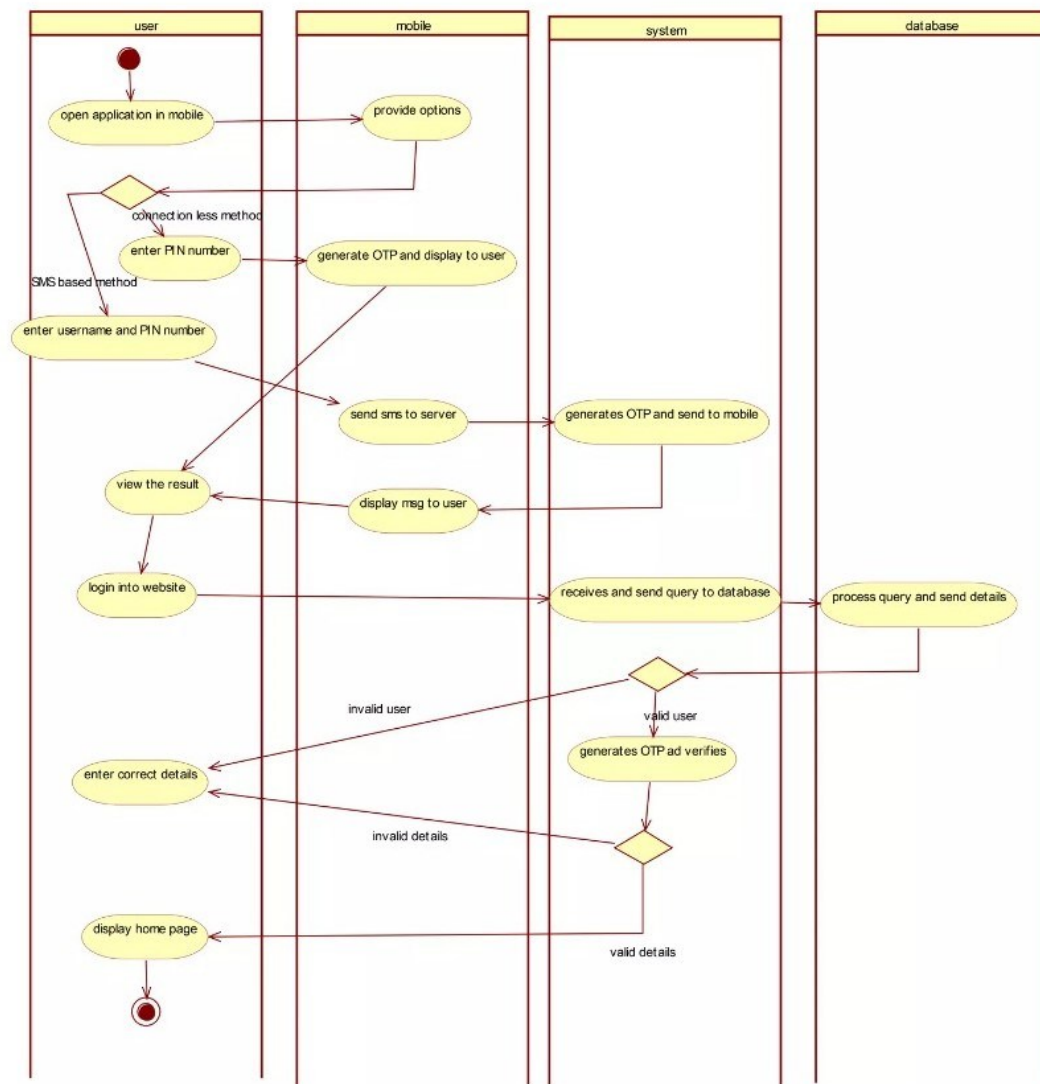


Fig 2.1: Activity Diagram of OTP Verification System

In an activity diagram for OTP verification, the process starts with the user requesting OTP generation. The system generates and sends the OTP via chosen means. The user inputs the OTP for verification. The system checks its validity; if correct, access is granted; if incorrect, access is denied with appropriate feedback shown as shown in Fig 2.1.

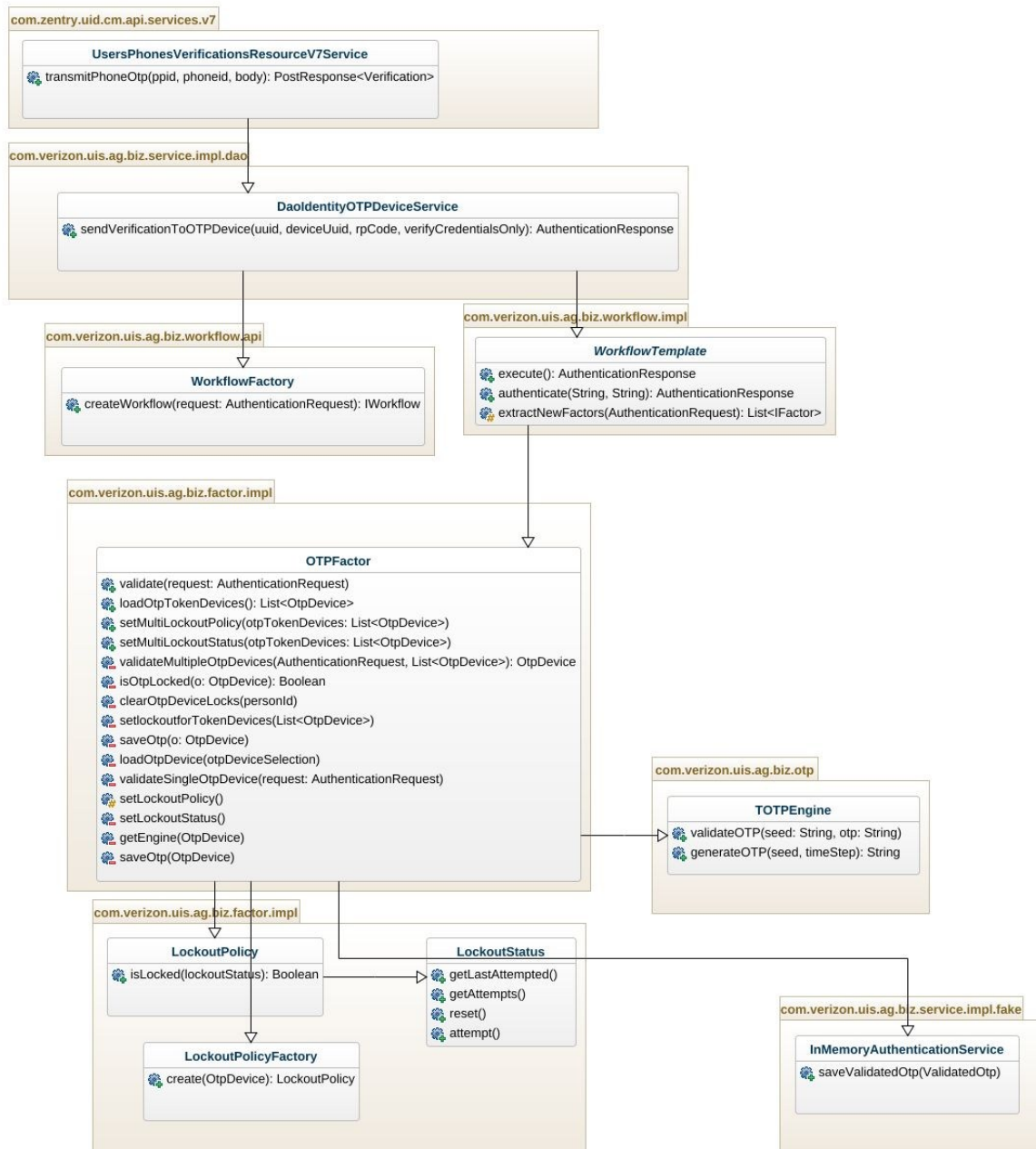


Fig 2.2: Class Diagram of OTP Verification System

In a class diagram for an OTP verification system, key classes include User, OTP Generator, OTP Sender, OTP Verifier, and Access Manager. User interacts with OTP Generator to generate OTPs, OTPSender to dispatch them, and OTPVerifier to validate entries. Access Manager controls access based on verification status, ensuring secure authentication processes as shown in Fig 2.2.



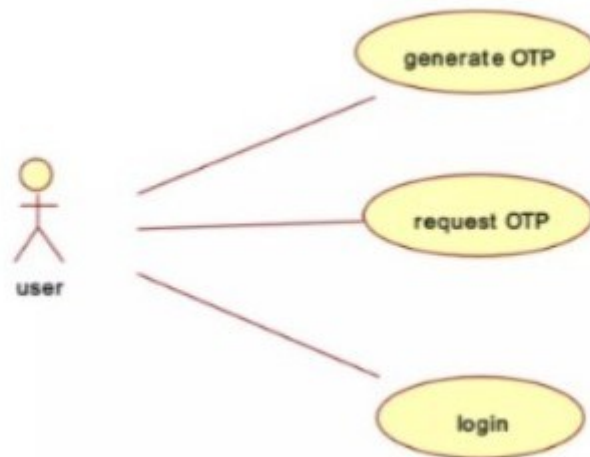


Fig 2.3: Use Case Diagram of OTP Verification System

A use case diagram for an OTP verification system illustrates actors like User and Admin interacting with use cases such as "Request OTP," "Verify OTP," and "Manage OTP Settings." Actors trigger OTP generation, input OTPs for verification, and manage system settings. It outlines how users interact with OTP functionalities within the system as shown in Fig 2.3

## **METHOD/TECHNIQUE/ALGORITHM**

Set up your server environment with necessary libraries and modules for email sending and OTP generation.

### **1.User Email Submission**

The user submits their email address to your system through a user interface.

### **2.OTP Generation**

Generate a secure OTP using a cryptographic library or an OTP generation library. Ensure the OTP is of adequate length (typically 6-10 characters) and complexity (alphanumeric or numeric only).

### **3.OTP Storage**

Store the generated OTP in your database with a reference to the user's email and a timestamp for expiration (usually 5-10 minutes).

### **4.Email Dispatch**

Use an email service provider or SMTP server to send the OTP to the user's email address. The email should contain clear instructions on where and how to enter the OTP.

### **5.User OTP Entry**

Provide a user interface for the user to enter the received OTP.

### **6.OTP Verification Algorithm**

Retrieve the stored OTP from the database using the user's email as a reference. Compare the user-entered OTP with the stored OTP. Check if the current time is within the expiration period set during storage.

## **7.Verification Outcome**

If the entered OTP matches and is within the valid time frame, authenticate the user's action. If there's no match or if the OTP has expired, prompt an error message and offer to resend a new OTP.

## **8.Post-Verification**

After successful verification, proceed with the user's requested action (e.g., account activation, password reset). Invalidate the used OTP to prevent replay attacks.

## **9.Security Considerations**

Implement rate limiting to prevent brute force attacks. Use secure channels (SSL/TLS) for all communications. Log attempts and set up alerts for multiple failed verifications.

## **IMPLEMENTATION**

### **CODE**

```
import random

import smtplib

OTP = random.randint(100000,999999)    #generating a random 6-digit OTP

#setting up server

server = smtplib.SMTP('smtp.gmail.com',587)

#server = smtplib.SMTP('64.233.184.108',587)    #IP address of smtp.gmail.com to
bypass DNS resolution

server.starttls()

name = input("enter your name:")

global receiver_email

receiver_email = input("enter ur email id:")

def email_verification(receiver_email):

    email_check1 = ["gmail","hotmail","yahoo","outlook"]

    email_check2 = [".com",".in",".org",".edu",".co.in"]

    count = 0

    for domain in email_check1:

        if domain in receiver_email:

            count+=1

    for site in email_check2:

        if site in receiver_email:

            count+=1

    if "@" not in receiver_email or count!=2:

        print("invalid email id")

        new_receiver_email = input("enter correct email id:")

        email_verification(new_receiver_email)

        return new_receiver_email

    return receiver_email
```

```

valid_receiver_email = email_verification(receiver_email)
password = "stqqwjqoocucknsx"
server.login("priyanshu25122002@gmail.com",password)
body = "dear"+name+", "+"\\n"+"\\n"+"your OTP is "+str(OTP)+"."
subject = "OTP verification using python"
message = f'subject: {subject}\\n\\n{body}'
server.sendmail("priyanshu25122002@gmail.com",valid_receiver_email,message)
def sending_otp(receiver_email):
    new_otp = random.randint(100000,999999)
    body = "dear"+name+", "+"\\n"+"\\n"+"your OTP is "+str(new_otp)+"."
    subject = "OTP verification using python"
    message = f'subject: {subject}\\n\\n{body}'
    server.sendmail("priyanshu25122002@gmail.com",receiver_email,message)
    print("OTP has been sent to"+receiver_email)
    received_OTP = int(input("enter OTP:"))
    if received_OTP==new_otp:
        print("OTP verified")
    else:
        print("invalid OTP")
        print("resending OTP.....")
        sending_otp(receiver_email)
    print("OTP has been sent to "+valid_receiver_email)
received_OTP = int(input("enter OTP:"))
if received_OTP==OTP:
    print("OTP verified")
else:
    print("invalid OTP")
    answer = input("enter yes to resend OTP on same email and no to enter a new email id:")
    YES = ['YES','yes','Yes']
    NO = ['NO','no','No']

```

```

if answer in YES:
    sending_otp(valid_receiver_email)
elif answer in NO:
    new_receiver_email = input("enter new email id:")
    email_verification(new_receiver_email)
    sending_otp(new_receiver_email)
else:
    print("invalid input")
server.quit()

```

## **RESULTS**

```








=====
enter your name:sai|

enter ur email id:saiteja.b222@gmail.com|
enter OTP:221339
OTP verified

enter your name:sai
enter ur email id:saiteja.b222@gmail.com
enter OTP:425809
invalid OTP
enter yes to resend OTP on same email and no to enter a new email id:yes
OTP has been sent tosaiteja.b222@gmail.com
enter OTP:289814
OTP verified
OTP has been sent to saiteja.b222@gmail.com




```


## SCREEN SHOT WITH DISSCUSSION



1 of 293


OTP verification using python Inbox x






**priyanshu25122002@gmail.com**  
dearsai, your OTP is 221339.

7:45 PM (8 minutes ago) ☆




**priyanshu25122002@gmail.com**  
dearsai, your OTP is 452089.


7:46 PM (7 minutes ago) ☆




**priyanshu25122002@gmail.com**  
to ▾  
dearsai,  
  
your OTP is 289814.

7:46 PM (6 minutes ago) ☆ 😊 ↩ ⋮

 Reply

 Forward



## **CONCLUSION**

Implementing an OTP (One-Time Password) verification system through email ensures secure user authentication for various online transactions. Upon user initiation, a unique OTP is generated and securely sent to their registered email address. The system stores the OTP hashed and salted, alongside an expiry timestamp to mitigate replay attacks. Users input the OTP into the application or website, triggering verification where the system compares it with the stored OTP and checks its validity period. Successful verification grants access to the requested action, while invalid or expired OTPs prompt users to retry or request a new OTP. Security measures include encrypted transmission, OTP expiration within minutes, rate limiting, and robust logging for auditing. User experience is enhanced through clear instructions and error-handling mechanisms, ensuring reliability across diverse scenarios. Continuous testing and user feedback drive iterative improvements, ensuring the OTP verification system remains resilient and user-friendly over time. Thus, concluding the implementation of an email-based OTP verification system guarantees both security and usability in online interactions.



## **REFERENCES**

- [1] S. Hallsteinsen, I. Jorstad, D-V., Thanh, “Using the mobile phone as a security token for unified authentication”, Systems and Networks Communication. In: International Conference on Systems and Networks Communications, 2007, pp. 68-74.
- [2] T. Laukkanen, S. Sinkkonen, M. Kivi Jarvi, P. Laukkanen, “Management of Mobile Business”, ICMB 2007, International Conference on the Digital Object Identifier, 2007, pp.42-42.
- [3] H. Wang, “Research and Design on Identity Authentication System in Mobile-Commerce”, In: Beijing Jiao Tong University, 2007, pp. 18-50.
- [4] S.M. Siddique, M. Amir, “GSM Security Issues and Challenges Software Engineering”, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2006. SNPD 2006. 7th ACIS International Conference on Digital Object Identifier, pp. 413-418.
- [5] L. Lamport, “Password Authentication with Insecure Communication”, In Comm. ACM, vol. 24, No 11, 1981, pp. 770-772.
- [6] N. Haller, “The S/KEY One–Time Password System. In: Proceedings of the ISOC Symposium on Network and Distributed System Security”, 1994, pp. 151-157.
- [7] K. Bicakci N. Baykal, “Infinite length hash chains and their applications” In: Proceedings of 1st IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborating Enterprises WETICE’02, 2002, pp. 57-61.
- [8] R. Rivest, A. Shamir, L. Adleman, “A method for obtaining digital signatures and public–key cryptosystems”, In: Communications of the ACM, 1978.
- [9] <http://www.rsa.com/node.aspx?id=1156>. [Accessed: October 04,2010].
- [10] A. Menezes, P. Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, Inc. 1997
- [11] L. Raddum, Nestås, K. Hole, “Security Analysis of Mobile Phones Used as OTP

Generators”, In IFIP International Federation for Information Processing.