SecureWhisper

Where Every Whisper is a Shield of Trust

PROJECT REPORT ON

SecureWhisper

UNDER THE GUIDANCE OF

Mrs. Krupa Kamdar

SUBMITTED BY

Mahika Gupta

UNIVERSITY OF MUMBAI

T.Y.B.Sc (COMPUTER SCIENCE)UNDER THE

ACADEMIC YEAR: 2023-2024

BHARTIYA VIDYA BHAVAN'S

MM COLLEGE OF ARTS, NM COLLEGE OF COMMERCE &

HRJ COLLEGE OF COMMERCE

BHAVAN'S COLLEGE

MUNSHI NAGAR, ANDHERI (WEST),

MUMBAI – 400058

BHARTIYA VIDYA BHAVAN'S
MM COLLEGE OF ARTS, NM COLLEGE OF COMMERCE &
HRJ COLLEGE OF COMMERCE
BHAVAN'S COLLEGE
MUNSHI NAGAR, ANDHERI (WEST),
MUMBAI – 400058

BHARATIYA VIDYA BHAVAN
BHAVAN'S COLLEGE, ANDHERI (W)

CERTIFICATE

This is to certify that **Ms Gupta Mahika Santosh** of the class **T. Y. B. Sc. Computer Science** has satisfactorily completed the practical course in **BH.USCS 606** as prescribed by the University of Mumbai during the academic year **2023-24**

Signature
Staff In Charge

Signature
Computer Science Coordinators

Signature
Examiner

College Stamp

ACKNOWLEDGEMENT

I would like to express my sincere gratitude towards the Computer Science Department of Bhavan's College.

After months of hard work, finally I am delighted to present my project. The Project making was full of new experiences, learning and difficult one too. Though a difficult job it was made simpler by the timely guidance received, which helped me greatly in the completion of my project. But it wouldn't be right to do so without thanking to those who have helped me in converting our thought into reality. So I would like to take full advantage of this opportunity to thank each and every person who has helped me throughout the completion of our project.

I am obliged to my parents & family members who always support me greatly and encouraged me in each and every step. I give my special thanks and sincere gratitude towards the Principal Dr. Zarin Bathena , Head of SFC Prof.[name] and Head of Department(Computer Science) Prof. Mrs. Krupa Balsara Kamdar.

I owe my sincere thanks to our Project guide Prof Ms. Nancy Nelson for their constant support and encouragement without which the successful completion of this project would have been impossible. They have been instrumental for making me concentrate and focus my effort in this project. And last but not the least the entire college staff.

I would like to give thanks to my client Mr. Santosh Gupta_for helping me in my project. Finally I would like to thank each and every individual who directly or indirectly contributing for this project**.**

Thank-you

INDEX

1] Abstract and keywords:

Abstract:

In response to the growing concerns over privacy and security in digital communication, this paper presents SecureWhisper, a cutting-edge chat system designed to safeguard user confidentiality through advanced encryption techniques. SecureWhisper utilizes the AES (Advanced Encryption Standard) algorithm, a widely recognized and trusted encryption method, to ensure that messages exchanged within the platform are protected from unauthorized access. By integrating AES encryption into its core functionality, SecureWhisper offers users a secure environment for communication, where sensitive information and private conversations can be shared with confidence. This paper outlines the architecture and features of SecureWhisper, emphasizing its ability to enable secure messaging across various devices and platforms while maintaining user privacy as a top priority.

Keywords:

Chat system, Encryption, Decryption, AES algorithm, Privacy, Security, Confidentiality, Secure communication, Advanced Encryption Standard, Digital privacy, Information security, Communication platform, Cybersecurity.

2] Introduction:

2.1) Problem Statement:

In the era of pervasive digital communication, ensuring the confidentiality and integrity of messages exchanged over messaging platforms has become increasingly imperative. However, existing messaging systems often lack robust security measures, leaving user communications vulnerable to interception, eavesdropping, and unauthorized access. Therefore, there is a pressing need for a messaging system that empowers users to encrypt and decrypt messages securely, thus safeguarding their privacy and ensuring that sensitive information remains protected from potential adversaries. This system should implement end-to-end encryption techniques, robust key management protocols, and user authentication mechanisms to establish secure communication channels while maintaining a seamless and user-friendly experience across various platforms.

2.2) Description of Present System:

The present messaging system facilitates communication between users through text-based messages exchanged over digital platforms. However, the system lacks comprehensive security measures to protect user communications from unauthorized access and interception. Currently, messages are transmitted in plain text format, leaving them vulnerable to eavesdropping and interception by malicious entities. Without encryption, sensitive information shared within messages is susceptible to exploitation, potentially compromising user privacy and confidentiality. Moreover, the absence of encryption mechanisms means that there are no safeguards in place to prevent unauthorized parties, including service providers or malicious actors, from accessing and tampering with message content during transmission. Additionally, the lack of robust key management protocols makes it challenging to securely exchange encryption keys between users, further compromising the system's security.

Overall, the present messaging system falls short in providing the necessary security measures to protect user communications adequately. To address these shortcomings, there is a critical need to enhance the system with robust encryption, key management, and authentication mechanisms to ensure the confidentiality, integrity, and privacy of messages exchanged between users.

2.3) Limitations

1. Lack of End-to-End Encryption: Many traditional messaging systems transmit messages in plain text, leaving them vulnerable to interception and eavesdropping by unauthorized parties. Without end-to-end encryption, sensitive information shared within messages is at risk of being compromised.
2. Vulnerability to Man-in-the-Middle Attacks: Without proper encryption and authentication mechanisms, normal messaging systems are susceptible to man-in-the-middle attacks, where an attacker intercepts and potentially alters the communication between two parties without their knowledge.
3. Limited Privacy Protection: Normal messaging systems often collect user data and metadata, such as message content, timestamps, and sender/receiver information, which can compromise user privacy. This data may be vulnerable to unauthorized access or surveillance by service providers or third parties.
4. Lack of Message Integrity Verification: In the absence of cryptographic hashing or other integrity verification mechanisms, normal messaging systems cannot guarantee the authenticity and integrity of messages. This leaves users vulnerable to message tampering or manipulation by malicious actors.

2.4) Aim & Objective:

Aim:

The aim of this project is to conceptualize, design, and implement a highly secure and user-friendly messaging system that prioritizes the encryption and decryption of messages. This system aims to revolutionize the way users communicate by providing an environment where the confidentiality and integrity of messages are paramount. By focusing on end-to-end encryption as a fundamental principle, the aim is to create a platform where users can communicate freely without fear of unauthorized access or interception of their sensitive information.
The aim is to empower users with the ability to encrypt their messages on their own devices before transmission, ensuring that only the intended recipients possess the necessary decryption keys to access the content. Through the implementation of robust encryption algorithms and protocols, the system aims to establish a secure communication channel that is resistant to interception, eavesdropping, and tampering.

Objectives:

1. Implement End-to-End Encryption: Develop and integrate robust encryption algorithms and protocols into the messaging system to enable end-to-end encryption of messages. This objective entails ensuring that messages are encrypted on the sender's device using strong encryption techniques and remain encrypted throughout transmission until they are decrypted by the intended recipient.
2. User-Friendly Interface: Design a user-friendly interface that simplifies the encryption and decryption process, making it accessible to users with varying levels of technical expertise. This objective involves creating intuitive interfaces for encrypting, decrypting, and managing messages, thereby enhancing user experience and adoption.
3. Performance and Scalability: Develop a scalable and high-performance system architecture capable of handling large volumes of encrypted messages while maintaining responsiveness and reliability. This objective aims to ensure that the messaging system can accommodate increasing user demand without sacrificing performance or security.
4. Privacy Preservation: Implement privacy-preserving practices to minimize the collection and storage of user data and metadata, ensuring user privacy and confidentiality of communications. This objective involves adhering to privacy principles such as data minimization, encryption of user data at rest, and transparent data handling practices to protect user privacy.

2.5) Project Motivation:

The motivation behind developing a messaging system that empowers users to encrypt and decrypt messages stems from the growing concerns surrounding digital privacy, security breaches, and unauthorized surveillance. In today's interconnected world, where communication plays a central role in both personal and professional spheres, ensuring the confidentiality and integrity of sensitive information exchanged online has become paramount. However, traditional messaging systems often fall short in providing adequate security measures to protect user communications from potential threats. The motivation for this project is rooted in the recognition of the pressing need to address these security shortcomings and provide users with a platform that prioritizes their privacy and security. By enabling end-to-end encryption of messages, the proposed messaging system seeks to restore user trust and confidence in digital communication channels. The motivation also arises from a desire to empower users with greater control over their personal data and communications, allowing them to communicate freely without fear

3] Description of Proposed Work

3.1) Number of Modules:

- Login
- Sign-up
- Search users
- Follow/Unfollow
- Encrypt the message
- Decrypt the message

3.2) Algorithm:

The algorithm used for encryption and decryption is AES.
The AES encryption algorithm operates through a series of rounds, the number of which depends on the key size. In each round, the plaintext undergoes a series of transformations using a set of round keys derived from the original encryption key. The process begins with an initial round where the plaintext block is XORed with the first round key. Subsequent rounds consist of four main operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey.
In the SubBytes step, each byte of the state is replaced with a corresponding byte from a substitution box (S-box), which is a predefined table of values. The ShiftRows step cyclically shifts the rows of the state matrix to introduce diffusion. MixColumns operates on each column of the state, applying a linear transformation to combine the bytes in a column. Finally, AddRoundKey XORs the state with the round key. These operations are repeated for a number of rounds determined by the key size, after which the encrypted plaintext is obtained. The strength of AES lies in its combination of these operations, which together provide robust encryption suitable for a wide range of applications.

3.3) Working

Sign-up:

Upon signing up, users will be prompted to create a unique username and password. Users will then have the option to log in using their credentials to access the system's features and functionalities.

Login:

Users can log in using their unique username and password. The system will verify their credentials and grant them access to the features and functionalities of the system. Once logged in, users can proceed to perform actions such as searching for other users, following or unfollowing them, and sending encrypted messages.

Search users:

The module "Search users" allows users to easily find specific individuals within the system.

Follow the user:

Once logged in, users can search for other users by their unique usernames. They can click the "follow" button on a user's profile page to start following them. Once they follow a user, they can send them an encrypted message.

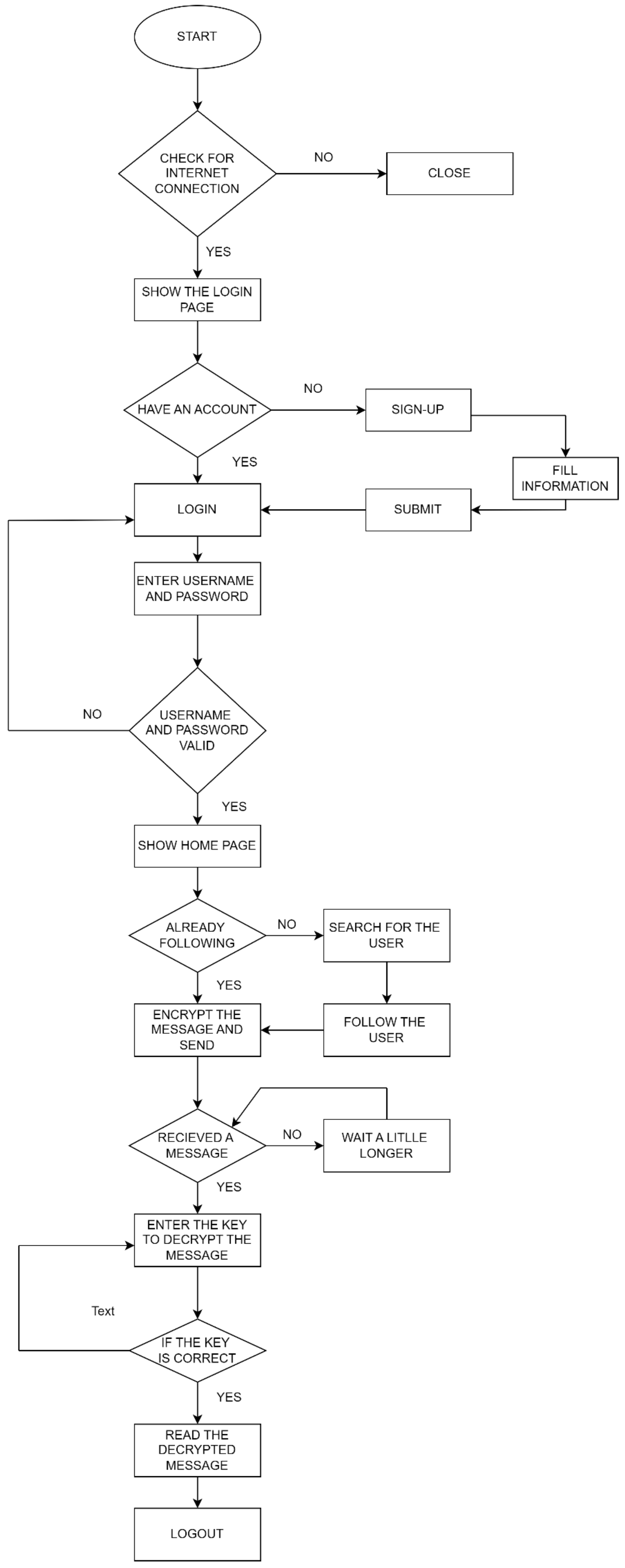
Encrypt and send the message:

It encrypts the message using AES, providing strong encryption and protection against unauthorized access. Once encrypted, the message can be safely sent to the intended recipient, ensuring that confidentiality is maintained throughout the communication process.

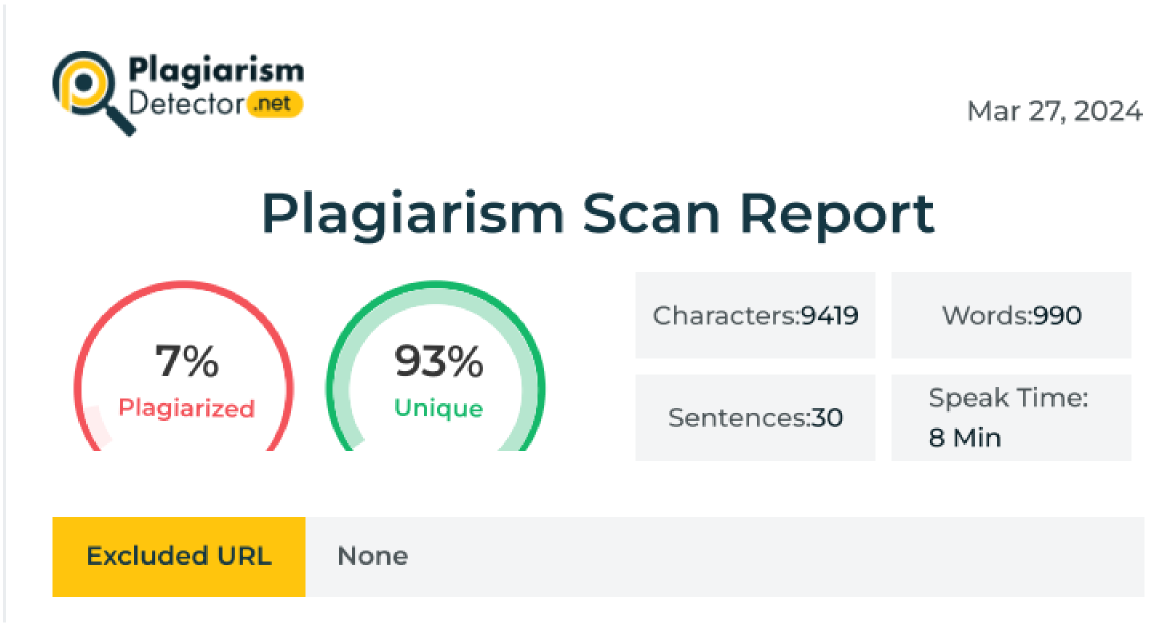
Enter the key and decrypt the message:

Upon receiving the encrypted message, the recipient can enter the decryption key into the system. The system will then use the AES algorithm to decrypt the message, ensuring that only authorized individuals can access the contents.

3.4) Flowchart:



3.5) Plagiarism report


Plagiarism Detector .net

Mar 27, 2024

Plagiarism Scan Report

7% Plagiarized

93% Unique

Characters:9419

Words:990

Sentences:30

Speak Time: 8 Min

Excluded URL None

3.6) Coding

views.py

```
from django.shortcuts import redirect, render, get_object_or_404,
HttpResponse
from django.contrib.auth.decorators import login_required
from directs.models import Message
from django.contrib.auth.models import User
from authy.models import Profile
from django.db.models import Q
from django.core.paginator import Paginator

from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from base64 import b64encode
from base64 import b64decode

@login_required
def inbox(request):
    user = request.user
    messages = Message.get_message(user=request.user)
    active_direct = None
    directs = None
    profile = get_object_or_404(Profile, user=user)

    if messages:
        message = messages[0]
        active_direct = message['user'].username
        directs = Message.objects.filter(user=request.user,
reciepient=message['user'])
        directs.update(is_read=True)

        for message in messages:
            if message['user'].username == active_direct:
                message['unread'] = 0
    context = {
        'directs':directs,
        'messages': messages,
        'active_direct': active_direct,
        'profile': profile,
    }
    return render(request, 'directs/direct.html', context)

def get_messages(request):
    m = request.POST.get('msg')
    k = request.POST.get('key')
    dec = decrypt_message(m,k)

    user = request.user
    messages = Message.get_message(user=request.user)
    active_direct = None
    directs = None
    profile = get_object_or_404(Profile, user=user)

    if messages:
        message = messages[0]
        active_direct = message['user'].username
        directs = Message.objects.filter(user=request.user,
reciepient=message['user'])
        directs.update(is_read=True)

        for message in messages:
            if message['user'].username == active_direct:
                message['unread'] = 0
    context = {
        'directs':directs,
        'messages': messages,
        'active_direct': active_direct,
        'profile': profile,
        'decrypteds': dec
    }
    return render(request, 'directs/direct.html', context)

    print(dec)
@login_required
def Directs(request, username):
    user  = request.user
    messages = Message.get_message(user=user)
    active_direct = username
    directs = Message.objects.filter(user=user,
reciepient__username=username)
    directs.update(is_read=True)

    for message in messages:
            if message['user'].username == username:
                message['unread'] = 0
    context = {
        'directs': directs,
        'messages': messages,
        'active_direct': active_direct,
    }
    return render(request, 'directs/direct.html', context)

def encrypt_message(message, key):
    cipher = AES.new(key.encode('utf-8'), AES.MODE_GCM)
    ciphertext, tag = cipher.encrypt_and_digest(message.encode())
    return b64encode(cipher.nonce + tag + ciphertext).decode('utf-8')

def decrypt_message(encrypted_message, key):
    encrypted_message_bytes =
b64decode(encrypted_message.encode('utf-8'))
    nonce = encrypted_message_bytes[:16]  # Nonce size for AES-GCM is
12 bytes (96 bits)
    tag = encrypted_message_bytes[16:32]  # Tag size for AES-GCM is 16
bytes
    ciphertext = encrypted_message_bytes[32:]

    # Create AES decryption cipher
    cipher = AES.new(key.encode('utf-8'), AES.MODE_GCM,
nonce=nonce)
```

```python
 # Decrypt and verify the ciphertext
    decrypted_message = cipher.decrypt_and_verify(ciphertext, tag)
    return decrypted_message.decode('utf-8')

def SendDirect(request):
    from_user = request.user
    to_user_username = request.POST.get('to_user')
    body = request.POST.get('body')
    key = request.POST.get('key')
    msg = encrypt_message(body,key)

    if request.method == "POST":
        to_user = User.objects.get(username=to_user_username)
        Message.sender_message(from_user, to_user, msg)
        return redirect('message')

def UserSearch(request):
    query = request.GET.get('q')
    context = {}
    if query:
        users = User.objects.filter(Q(username__icontains=query))

        # Paginator
        paginator = Paginator(users, 8)
        page_number = request.GET.get('page')
        users_paginator = paginator.get_page(page_number)

        context = {
            'users': users_paginator,
            }

    return render(request, 'directs/search.html', context)

def NewConversation(request, username):
    from_user = request.user
    body = ''
    try:
        to_user = User.objects.get(username=username)
    except Exception as e:
        return redirect('search-users')
    if from_user != to_user:
        Message.sender_message(from_user, to_user, body)
    return redirect('message')
```

direct.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}

<head>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min
.css" rel="stylesheet"
        integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqy
l2QvZ6jIW3" crossorigin="anonymous">

    <style>
        body {
            margin-top: 20px;
        }

        .chat-online {
            color: #34ce57
        }

        .chat-offline {
            color: #e4606d
        }

        .chat-messages {
            display: flex;
            flex-direction: column;
            max-height: 800px;
            overflow-y: scroll
        }

        .chat-message-left,
        .chat-message-right {
            display: flex;
            flex-shrink: 0
        }

        .chat-message-left {
            margin-right: auto
        }
        .chat-message-right {
           flex-direction: row-reverse;
           margin-left: auto
        }

        .py-3 {
           padding-top: 1rem !important;
           padding-bottom: 1rem !important;
        }

        .px-4 {
           padding-right: 1.5rem !important;
           padding-left: 1.5rem !important;
        }

        .flex-grow-0 {
           flex-grow: 0 !important;
        }

        .border-top {
           border-top: 1px solid #dee2e6 !important;
        }
    </style>
</head>
<br><br>
<main class="conetent">
    <div class="container p-0">
        {% if decrypteds %}
        <div class="alert alert-warning alert-dismissible fade show my-0"
role="alert">
            Decrypted Message is <strong>{{decrypteds}}</strong>
            <button type="button" class="btn-close" data-bs-dismiss="alert"
aria-label="Close"></button>
        </div>
        {% endif %}

        <h1 class="h3 mb-3">Messages</h1>

        <!-- Modal -->
        <div class="modal fade" id="exampleModal" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
            <div class="modal-dialog">
                <div class="modal-content">
                    <div class="modal-header">
```

```
                        <h1 class="modal-title fs-5"
id="exampleModalLabel">Modal title</h1>
                        <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
                    </div>
                    <div class="modal-body">
                        <form action="/message/test/" method="post">
                            {% csrf_token %}
                            <input type="text" class="form" name="msg"
placeholder="Enter the Message">
                            <input type="text" class="form" name="key"
placeholder="Enter the key">
                            <div class="modal-footer">
                                <button type="button" class="btn btn-secondary"
data-bs-dismiss="modal">Close</button>
                                <button type="submit" class="btn btn-primary">Save
changes</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>

        <div class="card">
            <div class="row g-0">
                <div class="col-12 col-lg-12 col-xl-3 border-right chat-
messages">

                    <div class="px-4 d-none d-md-block">
                        <div class="d-flex align-items-center">
                            <div class="flex-grow-1">
                                <a href="{% url 'search-users' %}" class="btn btn-
success mt-4 mb-4">New Message</a>
                            </div>
                        </div>
                    </div>
                    {% for message in messages %}
                    <a href="{% url 'directs' message.user.username %}"
                        class="list-group-item list-group-item-action border-0 {%
if active_direct == message.user.username %}active{% endif %}">
                        <div class="badge bg-success float-right"></div>
                        <div class="d-flex align-items-start pb-2">
                    <img src="{{ message.user.profile.image.url }}"
class="rounded-circle mr-1" alt="img"
                      width="40" height="40">

                    <div class="flex-grow-1 ml-6">
                      <b>{{message.user.profile.first_name}}
{{message.user.profile.last_name}}</b>
                      <div class="small"><span class="fas fa-circle chat-
online"></span>
                        @{{message.user.username}}</div>
                    </div>
                  </div>
                </a>
                {% endfor %}

                <hr class="d-block d-lg-none mt-1 mb-0">
              </div>
              <div class="col-12 col-lg-7 col-xl-9">
                <div class="py-2 px-4 border-bottom d-none d-lg-block">
                  <div class="d-flex align-items-center py-1">
                    <div>
                      <button type="button" class="btn btn-primary" data-
bs-toggle="modal" data-bs-target="#exampleModal">
                        Decrypt Messages
                      </button>
                    </div>
                  </div>
                </div>

                <div class="position-relative">
                  <div class="chat-messages p-4">

                    {% for direct in directs %}
                    {% if direct.sender == request.user %}
                    <div class="chat-message-right pb-2">
                      <div>
                        <a href=""><img src="{{
direct.sender.profile.image.url }}"
                              class="rounded-circle mr-1" alt="img"
width="40" height="40"></a>
                        <div class="text-muted small text-nowrap mt-2"
                          style="font-size:10px; color: rgba(180, 180, 180,
0);">
```

```
                                        <p style="font-size:10px; color:
black;">{{direct.date|date:"d M, Y"}}</p>
                                    </div>

                                </div>
                                <div class="flex-shrink-1 bg-light rounded py-2 px-3
ml-3 mf-div">
                                    <!-- <div class="font-weight-bold mb-1">Sharon
Lessman</div> -->
                                    {{direct.body}}

                                </div>
                            </div>
                            {% else %}
                            <div class="chat-message-left pb-2">
                                <div>
                                    <a href=""><img src="{{
direct.sender.profile.image.url }}"
                                            class="rounded-circle mr-1" alt="img"
width="40" height="40"></a>
                                    <div class="text-muted small text-nowrap mt-2"
                                        style="font-size:10px; color: rgba(180, 180, 180,
0);">
                                        <p style="font-size:10px; color:
black;">{{direct.date|date:"d M, Y"}}</p>
                                    </div>

                                </div>
                                <div class="flex-shrink-1 bg-light rounded py-2 px-3
ml-3 mf-div">
                                    <!-- <div class="font-weight-bold mb-1">Sharon
Lessman</div> -->
                                    {{direct.body}}
                                    <!-- Button trigger modal -->
                                </div>
                            </div>
                            {% endif %}
                            {% endfor %}

                        </div>
                    </div>

                    <div class="flex-grow-0 py-3 px-4 border-top">
```

```
                        <form method="POST" action="{% url 'send-directs' %}"
id="send-msg-form">
                            {% csrf_token %}
                            <div class="input-group">
                                <input type="hidden" name="to_user" id=""
value="{{active_direct}}">
                                <input name="body" type="text" class="form-
control" placeholder="Type your message">
                                <input name="key" id="encKey" type="text"
class="form-control"
                                    placeholder="Type your key">
                                <button class="btn btn-primary"
type="submit">Send</button>
                            </div>
                        </form>

                    </div>

                </div>
            </div>
        </div>
    </div>
</main>

<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-
js.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-
js.js"></script>
<script
src="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"></s
cript>
<script
src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script
>
<script src="https://cdnjs.cloudflare.com/ajax/libs/aes-js/3.1.2/index.js"
    integrity="sha512-
5e17hebfdkOW/S7tdw5zGbA9eqD747Wz1S2VlkEXavOhdeW12JQHRi
9LPb7NfWZcIl9AK98wxPod/Df9PgdGIw=="
    crossorigin="anonymous" referrerpolicy="no-referrer"></script>

<script>
    function decryptMessage(encryptedMessage, key) {
        var encryptedBytes = aesjs.utils.hex.toBytes(encryptedMessage);
        // Extract the nonce, tag, and ciphertext
        var nonce = encryptedBytes.slice(0, 16); // Nonce size for AES-
GCM is 12 bytes (96 bits)
        var tag = encryptedBytes.slice(16, 32);   // Tag size for AES-GCM is
16 bytes
        var ciphertext = encryptedBytes.slice(32);

        // Create the AES decryption instance
        var aesCtr = new aesjs.ModeOfOperation.ctr(key, new
aesjs.Counter(nonce));

        // Decrypt the ciphertext
        var decryptedBytes = aesCtr.decrypt(ciphertext);

        // Convert the decrypted bytes to a string
        var decryptedText = aesjs.utils.utf8.fromBytes(decryptedBytes);

        return decryptedText;
    }


    const collection = document.getElementsByClassName("mf-div");
    const buttons = document.getElementsByClassName("mybtns");
    for (let i = 0; i < collection.length; i++) {
        buttons[i].addEventListener('click', () => {
            m = collection[i].textContent;
            alert(m)
            k = prompt("Enter the key: ")
            console.log(m)
            dec = decryptMessage(m, k);
            alert(dec);
        })
    }



</script>

</html>
{% endblock content %}
```

Base.html

```
{% load static %}

<!DOCTYPE html>
<html lang="en">

<head>
    <title>SecureWhisper</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="shortcut icon" href="assets1/favicon.svg" type="image/x-
icon">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.8.1/font/bootstrap-icons.css">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min
.css" rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqy
l2QvZ6jIW3" crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="{% static
'assets1/style.css' %}">
    <script src="{% static 'assets1/script.js' %}" defer></script>
</head>

<body>
    <header class="header">
        <nav class="header__content">
            <div class="header__buttons">
                <a href="{% url 'index' %}" class="header__home">
                    <img src="{% static "assets2/images/logo.png" %}"
style="    width: 85px;
                    height: 64px;
                " />
                </a>

                <button class="header__theme-button" title="Toggle Theme">
                    <svg class="header__theme-button-moon"
xmlns="http://www.w3.org/2000/svg" width="24" height="24"
fill="var(--text-dark)" viewBox="0 0 16 16">
```

```
                    <svg class="header__theme-button-sun"
xmlns="http://www.w3.org/2000/svg" width="24" height="24"
fill="var(--text-dark)" viewBox="0 0 16 16">
                </button>
            </div>

            <form action="{% url 'search-users' %}" method="GET">
                <div class="header__search">
                    <input type="text" placeholder="Search" name="q"
value="{{ request.GET.q }}">
                    <button class="btn btn-primary"
type="submit">Search</button>
                    <svg width="24" height="24" viewBox="0 0 24 24"
fill="none" xmlns="http://www.w3.org/2000/svg">
                </div>
            </form>
            <div class="header__buttons header__buttons--mobile">
                <a href="{% url 'newpost' %}">
                    <svg width="24" height="24" viewBox="0 0 24 24"
fill="none" xmlns="http://www.w3.org/2000/svg">
                        <rect x="3" y="3" width="18" height="18" rx="5"
stroke="var(--text-dark)" stroke-width="1.8"/>
                        <line x1="12.1" y1="6.9" x2="12.1" y2="17.1"
stroke="var(--text-dark)" stroke-width="1.8" stroke-linecap="round"/>
                        <line x1="6.9" y1="11.8" x2="17.1" y2="11.8"
stroke="var(--text-dark)" stroke-width="1.8" stroke-linecap="round"/>
                    </svg>
                </a>
                <a href="#">
                    <svg width="24" height="24" viewBox="0 0 24 24"
fill="none" xmlns="http://www.w3.org/2000/svg">
                </a>
                <a href="{% url 'message' %}">
                    <svg width="24" height="24" viewBox="0 0 24 24"
fill="none" xmlns="http://www.w3.org/2000/svg">
                        <path d="M5.81038 19.7478C5.83176 19.4539 5.70787
19.1681 5.

                </a>
            </div>

            <div class="header__buttons header__buttons--desktop">

                <a href="{% url 'index' %}">
```

```
                    <svg width="24" height="24" viewBox="0 0 24 24"
fill="none"
                </a>
                <a href="{% url 'message' %}">
                    <svg width="24" height="24" viewBox="0 0 24 24"
fill="none" xmlns="http://www.w3.org/2000/svg">
                    </a>
                </span>

                <a href="">
                    <button class="profile-button">
                        <div class="profile-button__border"></div>
                        <a href="{% url 'profile' request.user %}">
                            <div class="profile-button__picture">
                                <img src="{{request.user.profile.image.url}}"
alt="User Picture">
                            </div>
                        </a>
                    </button>
                </a>
            </div>
        </nav>
    </header>

    {% block content %}
    {% endblock content %}



            </section>
        </section>
    </main>

    <nav class="navbar">
        <a href="#" class="navbar__button">
            <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">

        </a>
        <a href="{% url 'search-users'  %}" class="navbar__button">
            <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
                <
        </a>
        <a href="#" class="navbar__button">
```

```
          </svg>
        </a>
        <a href="#" class="navbar__button">
          <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
            <
        </a>
        <a href="{% url 'profile' request.user %}" class="navbar__button
profile-button">
          <div class="profile-button__border"></div>
          <div class="profile-button__picture">
            <img src="{{request.user.profile.image.url}}" alt="User
Picture">
          </div>
        </a>
    </nav>
</body>
<script>
</html>
```

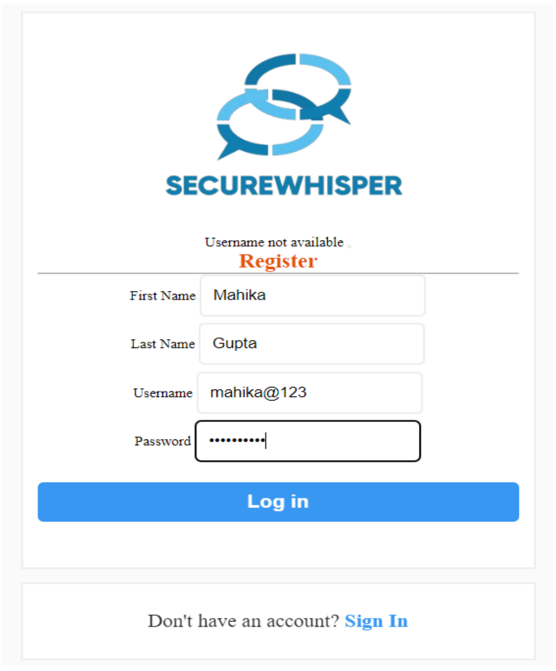
urls.py

```
from directs.views import inbox, Directs, SendDirect, UserSearch,
NewConversation, get_messages
from django.urls import path

urlpatterns = [
    path('', inbox, name="message"),
    path('direct/<username>', Directs, name="directs"),
    path('send/', SendDirect, name="send-directs"),
    path('search/', UserSearch, name="search-users"),
    path('new/<username>', NewConversation, name="conversation"),
    path('test/', get_messages, name="decry")
]
```

3.7) Screen Layouts

Signup


SECUREWHISPER

Username not available

Register

First Name: Mahika

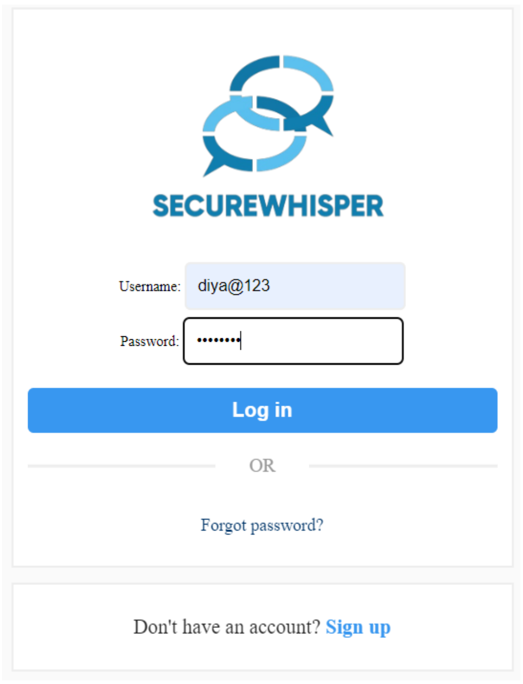
Last Name: Gupta

Username: mahika@123

Password: ••••••••••

Don't have an account? Sign In

Login

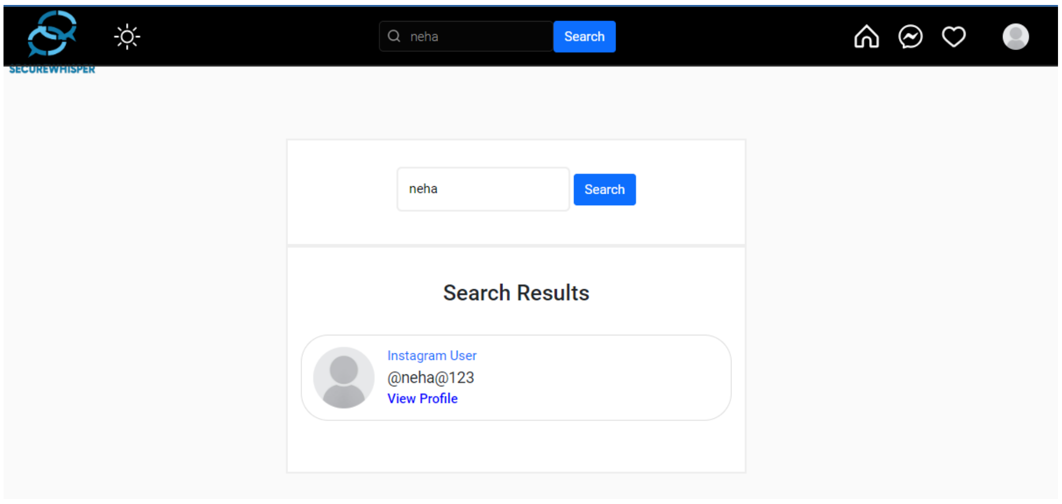

SECUREWHISPER

Username: diya@123

Password: ••••••••

Don't have an account? Sign up

Search users

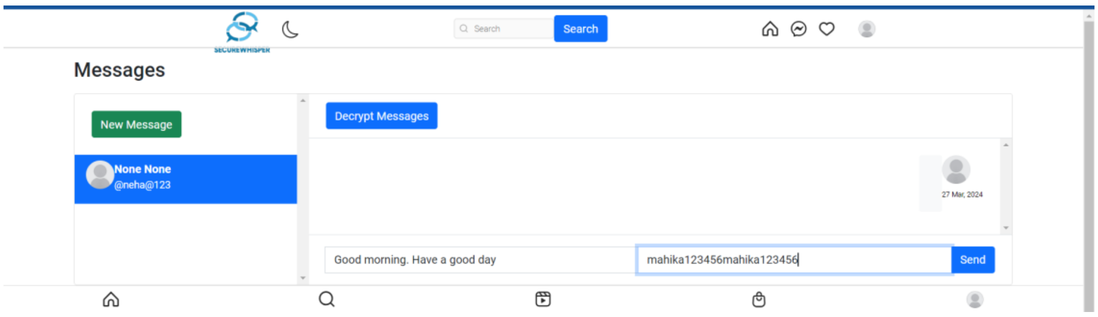


neha Search

Search Results

Instagram User
@neha@123
View Profile

Enter the message and the key



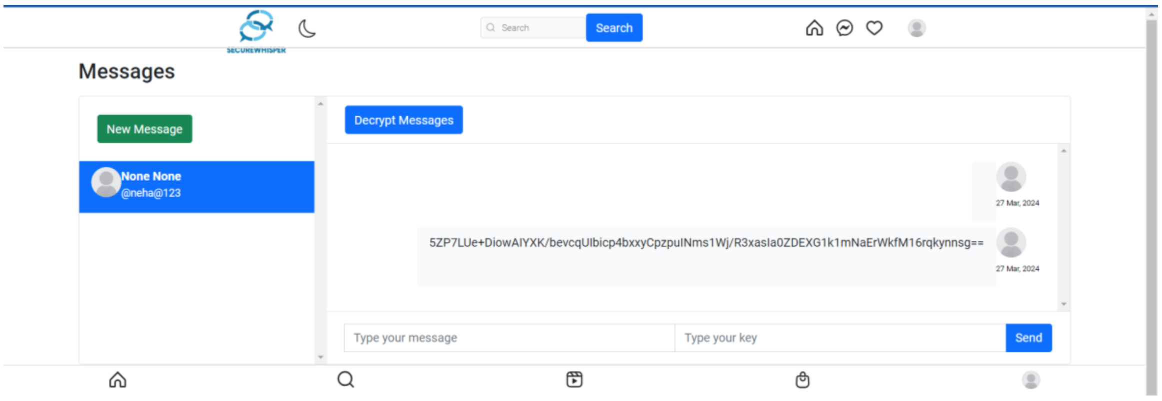
Messages

None None
@neha@123

27 Mar, 2024

Good morning. Have a good day | mahika123456mahika123456 | Send

The encrypted messages



None None
@neha@123

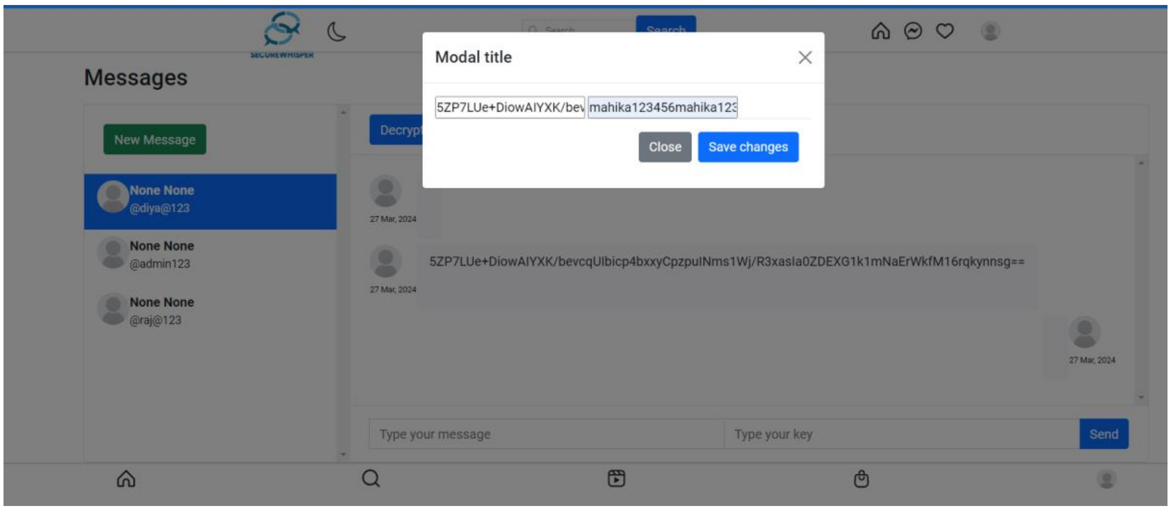
27 Mar, 2024

5ZP7LUe+DiowAIYXK/bevcqUIbicp4bxxyCpzpuINms1Wj/R3xasIa0ZDEXG1k1mNaErWkfM16rqkynnsg==

27 Mar, 2024

Type your message | Type your key

Decrypting the messages



Modal title

5ZP7LUe+DiowAIYXK/bev | mahika123456mahika123

None None
@diya@123

None None
@admin123

None None
@raj@123

27 Mar, 2024

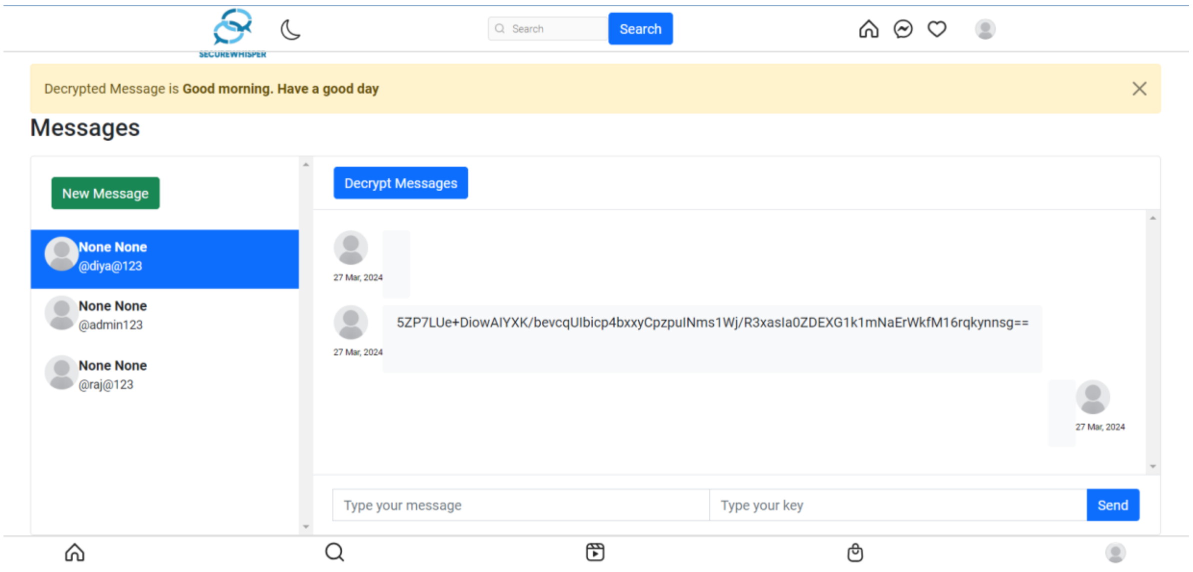
5ZP7LUe+DiowAIYXK/bevcqUIbicp4bxxyCpzpuINms1Wj/R3xasIa0ZDEXG1k1mNaErWkfM16rqkynnsg==

27 Mar, 2024

27 Mar, 2024

Type your message | Type your key

The decrypted message



Decrypted Message is **Good morning. Have a good day**

None None
@diya@123

None None
@admin123

None None
@raj@123

27 Mar, 2024

5ZP7LUe+DiowAIYXK/bevcqUIbicp4bxxyCpzpuINms1Wj/R3xasIa0ZDEXG1k1mNaErWkfM16rqkynnsg==

27 Mar, 2024

27 Mar, 2024

Type your message | Type your key

4] Technology used:

- Front End
  Bootstrap
- Back End
  Python Django,Sqlite

Software Requirements

a) Windows 7, 8, 10 or 11 with 32-bit or 64-bit architecture.
b) Microsoft Office
c) Google Chrome or Mozilla Firefox (Any web browser)

Hardware Requirements

a) 2 GB RAM or more
b) 1.6 GHz or fast processor
c) Intel i3 or more

5] Conclusion & Future Scope

Conclusion:

In conclusion, the development of SecureWhisper demonstrates the significant strides that can be made in enhancing user privacy and security within digital communication platforms. By leveraging the AES algorithm for message encryption and decryption, SecureWhisper provides users with a robust defense against potential threats to their confidentiality. Through this project, we have successfully addressed the pressing need for secure communication solutions in today's interconnected world, offering users a platform where they can engage in conversations with peace of mind, knowing that their messages are shielded from unauthorized access.

Future Scope:

Looking ahead, there are several avenues for further advancement and enhancement of SecureWhisper. One potential area of future exploration involves the integration of additional encryption algorithms to offer users a range of options for securing their messages. Moreover, ongoing research and development efforts can focus on improving the scalability and efficiency of SecureWhisper, ensuring seamless performance even as user demand grows. Additionally, incorporating features such as multi-factor authentication and end-to-end verification could further bolster the security measures of SecureWhisper, making it an even more robust solution for safeguarding user privacy in the digital realm. As technology continues to evolve and new challenges arise in the realm of cybersecurity, SecureWhisper stands poised to evolve and adapt, remaining at the forefront of secure communication solutions.

6] References/Resource Material/Data collection

- https://www.geeksforgeeks.org/advanced-encryption-standard-aes/
- https://www.javatpoint.com/aes-256-encryption-in-java
- https://winzip.com/blog/enterprise/aes-encryption-explained/#:~:text=Why%20Is%20AES%20the%20Preferred,level%20rather%20than%20bit%20level.
- https://www.cshub.com/mobile/news/cyber-security-risks-lurk-in-popular-messaging-apps#:~:text=Attacks%20can%20start%20with%20a,network%20and%20steal%20sensitive%20information.%E2%80%9D
- https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption