Web Chat Application

An internship project report submitted in
partial fulfillment of theRequirements for the
Summer training.

BACHELOR OF TECHNOLOGY

By

Adarsh Prakash

(BTECH/15136/18)

DEPARTMENT OF ELECTRONICS AND COMM. ENGG
BIRLA INSTITUTE OF TECHNOLOGY MESRA, PATNA
CAMPUS-80014

DECLARATION CERTIFICATE

This is to certify that the work presented in the thesis entitled

"Web Chat application"

in partial fulfilment of the requirement for the

"Summer Training"

of

"Birla Institute of Technology Mesra, Ranchi"

is an authentic work carried out under my supervision and guidance.

To the best of my knowledge, the content of this thesis does not form abasis for the award of any previous Degree to anyone else.

Date:

(Guide's Name & Signature)

Dept. of Electronics &Communication EngineeringBirla Institute of Technology

Mesra, Patna campus

Coordinator (Project)

(INCHARGE)

(Dept. of Electronics &Communication Engineering)Birla Institute of Technology

Mesra, Patna campus, 800014

CERTIFICATE OF APPROVAL

The foregoing thesis entitled "Web Chat Application",

is hereby approved as a creditable study of research topic andhas been presented in satisfactory manner to warrant its acceptance as prerequisite to the summer training for which ithas been submitted.

It is understood that by this approval, the undersigned do not necessarily indorse any conclusion drawn or opinion expressed there in, but approve the thesis for the purpose for which it is submitted.

(Internal Examiner) (External Examiner)

(Chairman)

Head of the Department

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning the people where ceaseless cooperation made it possible, whose constant guidance and encouragementcrown all the efforts with success.

We shall always be grateful to our parents whose moral support at every stepof our life is the reason for us to use ourselves. Their encouragement in everyjoy and belief of life laid the real foundation for our personality.

We sincerely thank our project guide Dr. Shiv Kumar Choubey (Assistant Professor, Electronics and Communication Engineering, Birla Institute of Technology, Mesra) for their great help, guidance, support, and ideas towards our project.

It was a great pleasure and good learning experience to have worked undertheir guidance during the tenure of the project.

ABSTRACT

This study begins with the development of a web application where one can chat with their friends privately or in a group chat. People can chat, share images, and even chat in other languages without translating it ourselves because the application does it for you.

Users can create rooms, which they can share with their friends and family and they can join those respective rooms.

So, the application is developed using python for the back-end, HTML, CSS, JavaScript for the frontend, and PostgreSQL for the database everything is discussed more in detail later in the thesis.

This chat app uses Socket Protocol for an end-to-end connection which doesn't require request methods, unlike HTTPS.

This report contains topics such as introduction, analysis, Advantages and Disadvantages, Design, Programming, and Conclusion.

CONTENTS

Introduction

In today's world of almost no virtual privacy and every tech company coming up with new tricks to collect more and more data from users, in some cases without the consent from the users. The situation of user privacy is at an all-time low. So, to combat this situation there needs to be an application where one can talk to their loved ones without worrying about their data and chat history potential being used by others.

A web chat application that doesn't save chat history can be a great platform for sharing your feelings without being obnoxious about your data privacy.

It is a web application that is written in python (for back-end) using the socket protocol for data transfer, Flask for web designing, HTML5, CSS, and JavaScript (for front-end), and PostgreSQL for the database.

The objective of this project was to develop a chat application that can be used for chatting and image sharing with the added benefits of privacy, and the academic objective was to learn Python, Flask framework, HTML, CSS, JavaScript, SQL, and a bit of server hosting.

Here, is the link of the deployed version of this web application.

And

Here, is the link of the GitHub repository of this project.

Features

- User account
  Users can create their accounts and then use the same id, password to log into the account from anywhere, on any device.

- Login Details encrypted
  Login details of the user are encrypted using the SHA-256 algorithm which produces a hash code that cannot be decrypted so even in the case of data leaks the user data is always protected.

- Private Chat rooms
  Users can create private chat rooms which they can then share with their friends and family and add them into the rooms and chat privately without bothering that someone unauthorized could potentially get their hands on their chat history.

- Image sharing
  Users can share images which opens another dimension through which they can share their feelings, emotions, and experiences

- Chat translation
  Users can now chat with people from other regions and languages with the use of real-time translation which saves them from the hassle of using other translation apps and then sending the message.

Programming Tools and Frameworks

Bank end:

- Python

Python is known for its simple syntax and short code length. This, paired with the fact that there is also extensive documentation. Python's countless resources come in many forms, including a wide variety of web application frameworks.
The biggest advantage of using Python is that it is the only programming language that offers a structure for the developers. For Python backend development, Django and Flask are the most popular python web frameworks.

- Why the Python Flask framework?

Flask is a web framework. Flask provides simplicity, flexibility, and fine-grained control. Flask allows you to build a web application by providing tools, libraries, and technologies. This web application will be a web page, a wiki, or a big web-based calendar application or commercial website. Flask is classified into a micro-framework that means it has little to no dependencies on external libraries.

Flask can handle HTTP requests easily with help of its functionalities

Flask uses templates to expand the functionality of a web application while maintaining a simple and organized file structure. Templates are enabled using the Jinja2 template engine and allow data to be shared and processed before being turned into content and sent back to the client

.

Front end

- HTML

HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and a Web page.

Hyper Text: HyperText simply means "Text within Text." A text that has a link within it, is a hypertext. Whenever you click on a link that brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages.

Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.

- CSS

CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language.

CSS helps Web developers create a uniform look across several pages of a Web site. Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document.

- JavaScript

JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. JavaScript can function as both a procedural and an object-oriented language.

JavaScript is a very powerful client-side scripting language. It is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage livelier and more interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development.

Database

- PostgreSQL

  - To handle the database of the application PostgreSQL has been used. The main reason was that it was a DBMS that was well supported by the server on which the application was deployed and it is also designed to handle many concurrent users at a time, which is the most important thing when building a chat application.

  - PostgreSQL, also known as Postgres, is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance.

- PostgreSQL features transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. It is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users.

- Postgres is added to our application using SQLALchemy, which is a python library that facilitates the communication between Python applications and databases. One of the biggest advantages of using SQLAlchemy is that it allows us to abstract all of our database logic into Python objects. Instead of having to think on a table, row, and column level, we can consider everything on a class, instance, and attribute level.

Server

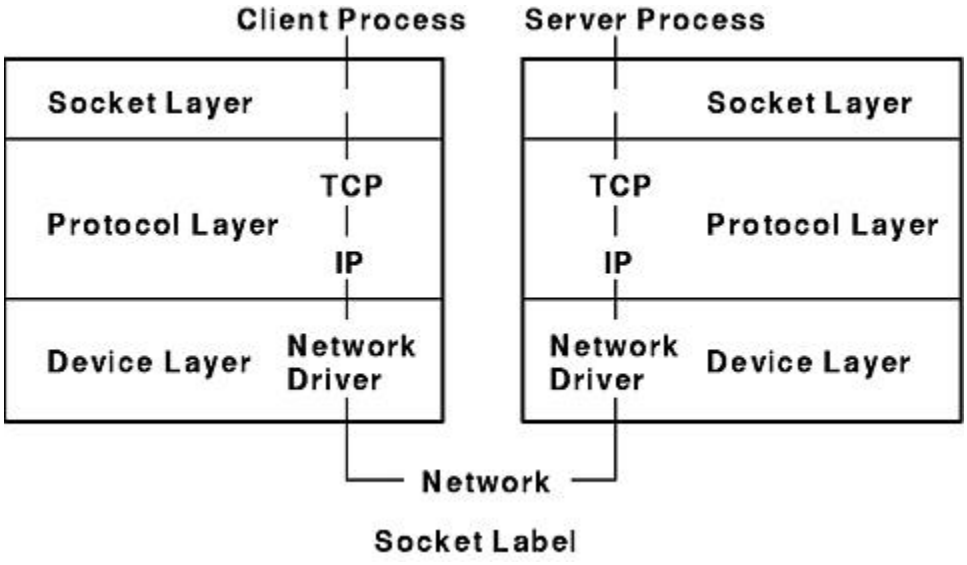
- Gunicorn

Gunicorn 'Green Unicorn' is a Python WSGI HTTP Server for UNIX. It's a pre-fork worker model. The Gunicorn server is broadly compatible with various web frameworks, simply implemented, light on server resources, and fairly speedy.

Data Transfer Protocol

- Network socket

Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors. In Unix, every I/O action is done by writing or reading a file descriptor. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.



APIs

Google Translate API

Cloud Translation enables your websites and applications to dynamically translate text programmatically through an API. Translation uses a Google pre-trained or a custom machine learning model to translate text. By default, Translation uses a Google pre-trained Neural Machine Translation (NMT) model, which Google updates on semi-regular cadence when more training data or better techniques become available.

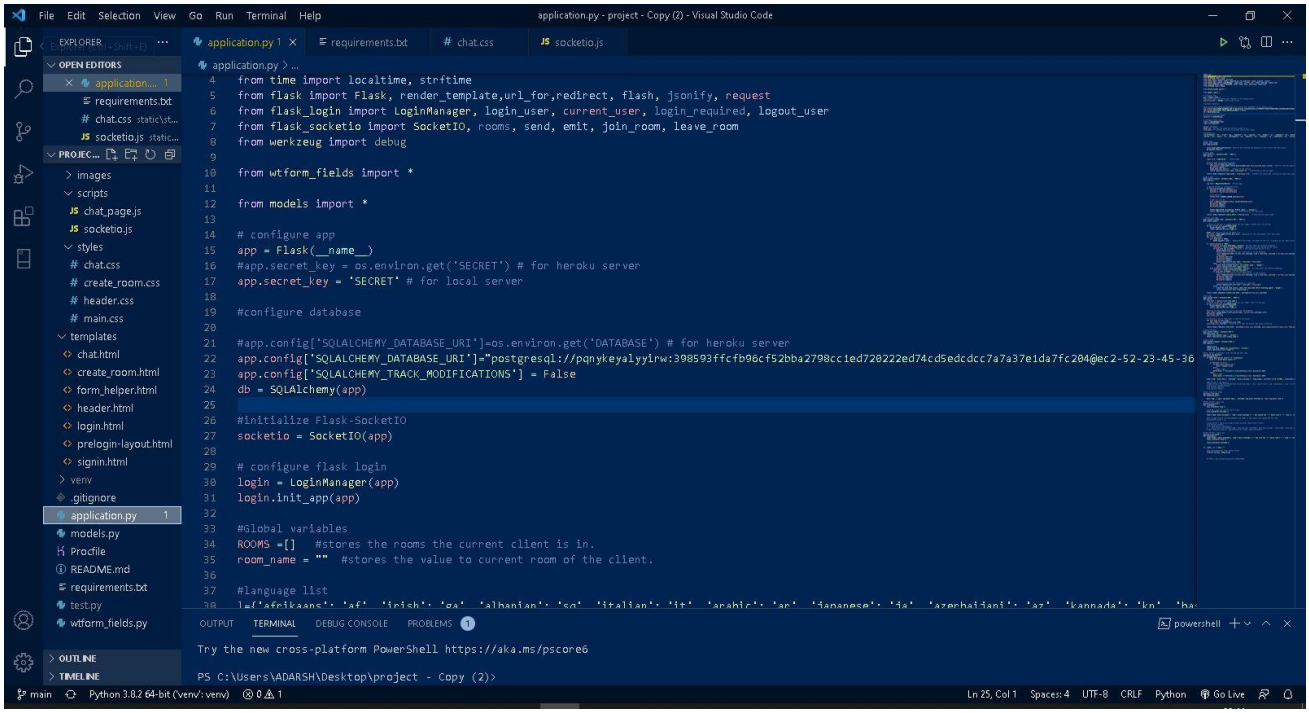
Software Requirement

- Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NETand Unity).

It is a source-code editor made by Microsoft for Windows, Linux and macOS.

Visual Studio Code in Action



Code

- application.py:

```
import os
from googletrans import Translator
from threading import local
from time import localtime, strftime
from flask import Flask, render_template,url_for,redirect, flash, jsonify, request
from flask_login import LoginManager, login_user, current_user, login_required, logout_user
from flask_socketio import SocketIO, rooms, send, emit, join_room, leave_room
from werkzeug import debug

from wtform_fields import *

from models import *

# configure app
app = Flask(__name__)
#app.secret_key = os.environ.get('SECRET') # for heroku server
app.secret_key = 'SECRET' # for local server

#configure database

#app.config['SQLALCHEMY_DATABASE_URI']=os.environ.get('DATABASE') # for heroku server
app.config['SQLALCHEMY_DATABASE_URI']="postgresql://pqnykeyalyyirw:398593ffcfb96cf52bba2798cc1ed720222ed74cd5edcdcc7
a7a37e1da7fc204@ec2-52-23-45-36.compute-1.amazonaws.com:5432/dam6i6c7a0u5i4" # for local server
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

#initialize Flask-SocketIO
socketio = SocketIO(app)

# configure flask login
login = LoginManager(app)
login.init_app(app)

#Global variables
ROOMS =[]   #stores the rooms the current client is in.
room_name = ""  #stores the value to current room of the client.

#language list
l={'afrikaans': 'af', 'irish': 'ga', 'albanian': 'sq', 'italian': 'it', 'arabic': 'ar', 'japanese': 'ja', 'azerbaijani': 'az', 'kannada': 'kn', 'basque': 'eu',
'korean': 'ko', 'bengali': 'bn', 'latin': 'la', 'belarusian': 'be', 'latvian': 'lv', 'bulgarian': 'bg', 'lithuanian': 'lt', 'catalan': 'ca', 'macedonian':
'mk', 'chinese(simplified)': 'zh-CN', 'malay': 'ms', 'chinese(traditional)': 'zh-TW', 'maltese': 'mt', 'croatian': 'hr', 'norwegian': 'no',
'czech': 'cs', 'persian': 'fa', 'danish': 'da',
'polish': 'pl', 'dutch': 'nl', 'portuguese': 'pt', 'english': 'en', 'romanian': 'ro', 'esperanto': 'eo', 'russian': 'ru', 'estonian': 'et', 'serbian': 'sr',
'filipino': 'tl', 'slovak': 'sk', 'finnish': 'fi', 'slovenian': 'sl', 'french': 'fr', 'spanish': 'es', 'galician': 'gl', 'swahili': 'sw', 'georgian': 'ka',
'swedish': 'sv', 'german': 'de', 'tamil': 'ta', 'greek': 'el', 'telugu': 'te', 'gujarati': 'gu', 'thai': 'th', 'haitian': 'ht', 'turkish': 'tr', 'hebrew': 'iw',
'ukrainian': 'uk', 'hindi': 'hi', 'urdu': 'ur', 'hungarian': 'hu', 'vietnamese': 'vi', 'icelandic': 'is', 'welsh': 'cy', 'indonesian': 'id'}
```

```
#flask login
@login.user_loader
def load_user(id):

   return User.query.get(int(id))  #returns the username and password of the client from users table.
   db.session.remove()

# login page
@app.route("/", methods=['GET', 'POST'])
def login():

   login_form = LoginForm()    #flask login

   # Allow login if validation success
   if login_form.validate_on_submit():
      user_object = User.query.filter_by(username=login_form.username.data).first()   #returns the user_object (ie. id, username,
password) of the clinet from the users table.
      db.session.remove()
      login_user(user_object)     #logins in the session
      return redirect(url_for('chat',room_name=""))    #redirecting to the cat page.

   return render_template("login.html", form=login_form)   #renders the login page, passing the login_form values as form variable
to html.

#signin page
@app.route("/signin", methods=['GET', 'POST'])
def signin():

   reg_form = RegistrationForm()   #flask login

   # updated databse if validation success
   if reg_form.validate_on_submit():
      username = reg_form.username.data
      password = reg_form.password.data

      #hash password
      hashed_pswd = pbkdf2_sha256.hash(password)

      # add user into db
      user = User(username=username, password=hashed_pswd)
      db.session.add(user)
      db.session.commit()
      db.session.remove()

      flash('Registered succesfully. Please login.', 'success')
      return redirect(url_for('login'))   #redireting to the login page.

   return render_template("signin.html", form=reg_form)    #rendering the signin page.

# Create_room page
@app.route("/create_room", methods=['GET', 'POST'])
def create_room():
```

```
    # checks if the user is logged in and not just added "/create_room" at the end
    if not current_user.is_authenticated:
        flash('Please login.', 'danger')
        return redirect(url_for('login'))

    ROOMS = []  #initializeing the ROOMS list.
    get_rooms = Rooms.query.filter_by().all()   #getting all the room objects from rooms table.
    db.session.remove()
    for i in get_rooms:
        if i.room not in ROOMS:
            ROOMS.append(i.room)    #adding all the unique room names to the list to display at the rooms section of the chat page.

    if request.method == 'POST':
        if (request.form.get('Room_name', False)):  #if the client has entered something.
            room_name = request.form['Room_name']   #getting the room entered by the client.
            if room_name not in ROOMS:  #checks if the room already exists or not.
                ROOMS.append(room_name)
                #Adding the room name the user connected to, to the database.
                room = Rooms(username =current_user.username, room = room_name, userroom = (current_user.username
                +room_name))
                db.session.add(room)
                db.session.commit()
                db.session.remove()
                return redirect(url_for('chat', room_name = room_name))
            else:   #if the room already exists.
                flash('Room Already Exists!, try another name.', 'danger')
                return redirect(url_for('create_room'))
        elif (request.form.get('join_room_name', False)):   #if the client has entered something.
            room_name = request.form['join_room_name']
            if room_name in ROOMS:  #if room already exists.
                #Adding the room name the user connected to, to the database.
                room = Rooms(username =current_user.username, room = room_name, userroom =
(current_user.username+room_name))
                db.session.add(room)
                db.session.commit()
                db.session.remove()

                # current_user.session['room_name'] = room_name
                return redirect(url_for('chat', room_name = room_name))
            else:   #if it did'nt.
                flash('No such room exists, check the room name before entering again!','danger')
                return redirect(url_for('create_room'))

    return render_template('create_room.html', username=current_user.username)

#chat page.
@app.route("/chat", methods=['GET', 'POST'])
def chat():
    room_name = request.args['room_name']
    # checks if the user is logged in and not just added "/chat" at the end
    if not current_user.is_authenticated:
```

```
        flash('Please login.', 'danger')
        return redirect(url_for('login'))

    #getting all the rooms this user is in from the database.
    user_rooms = Rooms.query.filter_by(username = current_user.username).all()
    db.session.remove()
    user_rooms_list = []

    # creating a list of user_rooms to send to the client.
    for user_room in user_rooms:
        user_rooms_list.append(user_room.room)
    user_rooms_list.reverse()   #reverse it so that the latest room comes in the top.

    return render_template('chat.html', username=current_user.username, user_rooms_list=user_rooms_list,
room_name=room_name)

#leave_room page
@app.route("/leave", methods=['GET'])
def leave_room__():
    flash("Join another room.", 'success')
    return redirect(url_for('create_room'))

#logout page
@app.route("/logout", methods=['GET'])
def logout():

    logout_user()
    flash('You have logged out successfully', 'success')
    return redirect(url_for('login'))

#event handler (socketIO): takes the msg and adds time.
@socketio.on('message')
def message(data):
    if data['msg'][0:10].lower() == "/translate":
        data_tr = data['msg'].split(":")

        if len(data_tr) == 3:
            if data_tr[1].lower() in l:
                dest = l[data_tr[1]]
            else:
                dest = 'en'
            data['msg'] = Translator().translate(data_tr[2], dest=dest).text
        else:
            dest = 'en'
            data['msg'] = Translator().translate(data_tr[1], dest=dest).text

    send({'msg': data['msg'], 'username': data['username'], 'time_stamp': strftime('%b-%d %I:%M%p', localtime())}, room
=data['room']) #msg={'msg':msg,'username':username}

    #adding msg to the database
    # msg_history = Msg_history(username =data['username'], room = data['room'], msg = data['msg'], time = strftime('%b-%d
%I:%M%p', localtime()))
```

```
    # db.session.add(msg_history)
    # db.session.commit()
    # db.session.remove()

#event handle for image
@socketio.on('img')
def image(img_data):

    emit('img', {'img': img_data['img'], 'username':img_data['username']}, room= img_data['room'])

#event handler: join room
@socketio.on('join')
def join(data):
    join_room(data['room'])

    #joining a seperate room for each client
    join_room(data['username'])

    send({'name':data['username'], 'msg': data['username'] + " has joined the '" + data['room'] + "' room."}, room=data['room'])

    #get message history from the database and sends to the client just joined the new room
    # msg_history_list = []

    # msg_history = Msg_history.query.filter_by(room = data['room']).all()
    # db.session.remove()
    # for each_msg in msg_history:
    #     msg_history_list.append({'msg': each_msg.msg, 'username': each_msg.username, 'time_stamp': each_msg.time})
    # emit('new_join_history', msg_history_list, room = data['username'])

#event handler: leave room
@socketio.on('leave')
def leave(data):
    send({'name': data['username'], 'msg': data['username'] + " has left the '" + data['room'] + "' room."}, room=data['room'])
    leave_room(data['room'])

    leave_room(data['username'])

if __name__ == "__main__":

    #app.run(debug=True) # for heroku server
    socketio.run(app, debug=True)
```

- **models.py:**

```
from enum import unique
from operator import truediv
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin

db = SQLAlchemy()

class User(UserMixin, db.Model):
    """ User model """

    __tablename__ = "users"
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(25), unique=True, nullable=False)
    password = db.Column(db.String(), nullable=False)

class Rooms(UserMixin, db.Model):
    """ rooms model """

    __tablename__ = "rooms"
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(25), nullable=False)
    room = db.Column(db.String(), nullable=False)
    userroom = db.Column(db.String(), unique=True, nullable=False)

class Msg_history(UserMixin, db.Model):
    """ msg_history model """

    __tablename__ = "msg_history"
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(25), nullable=False)
    room = db.Column(db.String(), nullable=False)
    msg = db.Column(db.String(), nullable=False)
    time = db.Column(db.String(), nullable=False)
```

- **wtform_fields.py:**

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import InputRequired, Length, EqualTo, ValidationError
from models import User

from passlib.hash import pbkdf2_sha256

def invalid_credentials(form, field):
    """ USername and password checker """

    username_entered = form.username.data
    password_entered = field.data

    #check user credentials is valid
    user_object = User.query.filter_by(username=username_entered).first()
    if  user_object is None:
        raise ValidationError("Username or password is incorrect!")
    elif not pbkdf2_sha256.verify(password_entered, user_object.password):
        raise ValidationError("Username or password is incorrect!")

class RegistrationForm(FlaskForm):
    """ Registration form """

    username = StringField('username_label', validators=[InputRequired(message="Username required"),
        Length(min=4, max=25, message="Username must be between 4 and 25 charaters")])
    password = PasswordField('password_label', validators=[InputRequired(message="Password required"),
        Length(min=4, max=25, message="Password must be between 4 and 25 charaters")])
    confirm_pswd = PasswordField('confirm_pswd_label', validators=[InputRequired(message="Password required"),
        EqualTo('password' , message="Password must match")])

    submit_button = SubmitField('Create')

    def validate_username(self, username):
        user_object = User.query.filter_by(username=username.data).first()
        if  user_object:
            raise ValidationError("Username already exists, Select a different username.")

class LoginForm(FlaskForm):
    """ LOGIN FORM """

    username = StringField('username_label',
        validators=[InputRequired(message="username required")])
    password = PasswordField('password_label',
        validators=[InputRequired(message="Password required"),
        invalid_credentials])
    submit_button = SubmitField('Login')
```

- **chat.html**

```
{% extends 'header.html' %}

{% block title%}Chat{% endblock %}

{% block content %}

   <div id="enlarged-img" class="enlarged-img">
      <i id="close-enlarged-img" class="fas fa-arrow-left fa-lg"></i>
      <img id="open_img" class="open_img">
      <i id="download-enlarged-image" class="fas fa-cloud-download-alt fa-lg"></i>
   </div>
   <p class="chat-img" id="hide"></p>
   <div id="main-section">
      <!--room selection-->
      <div class="transparent-bar mobile-hide"></div>
      <div id="sidebar" class="mobile-hide">
         <div class="sidebar_with_tilte"></div>
         <h4 class="rooms_text">Rooms</h4>
            <nav id="sidebar_rooms">
               {% for i in user_rooms_list %}
                  <h5 class="select-room" id="{{i}}">{{ i }}</h5>
                  <hr color="white">
               {% endfor %}
            </nav>
      </div>

      <!--message are-->
      <div id="rightside-panel">
         <div class="main-chat-section">

            <!--Display message start-->
            <div id="display-message-section">

            </div>
            <!--Display message ends-->

            <!--input message-->
            <div>
               <div class="selected_img" id="selected_img">
                  <p id="image_preview"><img id="img_output"><i id = "cancel-img" class="far fa-times-circle fa-lg"></i></p>
               </div>
               <input type="text" id="user_message" placeholder="Type here.." autocomplete="off" class="form-control form-
rounded" maxlength="150">
               <label for="img_upload" class="custom-img-upload">
                  <i class="fas fa-lg fa-file-image" id="send_image"></i>
               </label>
               <input id="img_upload" type="file" accept="image/*"/>
               <button class="btn btn-circle" type="button" id="send_message"><i class="fas fa-arrow-right"></i></button>
            </div>
         </div>
```

```
    </div>

  </div>
  <!-- get username ans room_name-->
  <script type="text/javascript">
    const username = `{{ username }}`;
    const room_name = `{{ room_name }}`;
    const user_rooms_list = `{{ user_rooms_list }}`
  </script>


{% endblock %}
```

- **Create_room.html:**

```
{% extends 'header.html' %}

{% block title%}Create_room{% endblock %}

{% block content %}
  <div class="background_create_room">
    <!--Flash message-->
    {% with messages = get_flashed_messages(with_categories=true) %}
       {% if messages %}
       <div class="alert alert-warning" role="alert">
         {{ messages[0][1] }}
       </div>
       {% endif %}
     {% endwith %}
    <div class="center_block">
      <div class="create_room">
        <form action = "{{ url_for('create_room') }}" method = "POST">
          <p class="text create_text">Create Your Own Room:</p>
          <div class="input-group input-group-sm mb-3">
            <input type = "text" name = "Room_name" placeholder="Enter a name" autocomplete="off" required="required"
class="form-control create_input">
          </div>
          <p><input type = "submit" value="Create" class="btn btn-outline-light create_button"></p>
        </form>
      </div>


      <div class="join_room">
        <p class="text join_text">Join a room:</p>
        <form action = "{{ url_for('create_room') }}" method = "POST">
          <div class="input-group input-group-sm mb-3">
            <input type = "text" name = "join_room_name" placeholder="Enter the room name" required="required" class="form-
control join_input">
          </div>
          <p><input type = "submit" value = "Join" class="btn btn-outline-light join_button"></p>
        </form>
      </div>
    </div>
  </div>
{% endblock %}
```

- **form_helper.html:**

```
{% macro displayField(fieldName, placeholderValue) %}

  <div>
    {{ fieldName(class_='form-control', placeholder=placeholderValue, **kwargs) }}

    <!-- Error messages-->

    <ul>
      {% for error in fieldName.errors %}
      <li>
        {{ error }}
      </li>
      {% endfor %}
    </ul>
  </div>
{% endmacro %}

Header.html:
<html>
  <head>
    <!--title image-->
    <link rel="apple-touch-icon" sizes="180x180" href="./static/images/apple-touch-icon.png">
    <link rel="icon" type="image/png" sizes="32x32" href="./static/images/favicon-32x32.png">
    <link rel="icon" type="image/png" sizes="16x16" href="./static/images/favicon-16x16.png">
    <link rel="manifest" href="./static/images/site.webmanifest">

    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale-1, shrink-to-fit=no">
    <title>{% block title%}{% endblock %}</title>
    <link rel="stylesheet" type="text/css" href="static\styles\header.css">
    <link rel="stylesheet" type="text/css" href="static\styles\create_room.css">
    <link rel="stylesheet" type="text/css" href="static\styles\chat.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="anonymous">
    <script src="https://kit.fontawesome.com/53136ede93.js" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></script>

    <!--custom js-->
    <script src="{{ url_for('static', filename='scripts/chat_page.js') }}"></script>

    <!-- SocketIO JS -->
    <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/socket.io.min.js"></script>

    <!-- Custom SocketIO JS -->
    <script src="{{ url_for('static', filename='scripts/socketio.js') }}"></script>

  </head>
  <body>
    <div class="background">
      <i class="fas fa-bars fa-lg show-mobile" id="ham"></i>
```

```
        <div class="top_bar show-mobile"></div>
        <nav class="navbar_left mobile-hide">
            <i class="fas fa-bars fa-lg show-mobile" id="ham_exit"></i>
            <img src="static\images\person.svg" class="user mobile-hide">
            <p class="user_name mobile-hide" >{{ username }}</p>
            <a id="create_room" href={{url_for('create_room')}} class="join_room_button mobile-hide"><img class="join_room_icon
mobile-hide" src="static\images\person-plus-fill.svg" alt="Join Room"></a>
            <a class="logout mobile-hide" id="logout_button" href={{url_for("logout")}}><img class="logout_icon"
src="static\images\box-arrow-right.svg"></a>
        </nav>
        <div>
            {% block content %}

            {% endblock %}
        </div>
      </div>
   </body>
</html>
```

- **Login.html:**

```
{% from 'form_helper.html' import displayField %}
{% extends 'prelogin-layout.html' %}

{% block title%}Login {% endblock %}

{% block content %}
  <div class="form">
    <h2 class="heading">Log In</h2>

    <form action="{{url_for('login')}}" method="POST">
      {{ displayField(form.username, 'Username', autocomplete="off", autofocus=true) }}
      {{ displayField(form.password, 'Password') }}

      <div class="form-group">
        <input type="submit" value="Login" class="btn btn-secondary">
      </div>

      {{ form.csrf_token }}

    </form>

    <hr class="hr">

    <p class="bottom_msg">Not Registered? <a href={{url_for('signin')}} class="link">Signup here</a>.</p>
  </div>

{% endblock %}
```

- **prelogin-layout.html:**

```
<html>
  <head>
    <!--title image-->
    <link rel="apple-touch-icon" sizes="180x180" href="./static/images/apple-touch-icon.png">
    <link rel="icon" type="image/png" sizes="32x32" href="./static/images/favicon-32x32.png">
    <link rel="icon" type="image/png" sizes="16x16" href="./static/images/favicon-16x16.png">
    <link rel="manifest" href="./static/images/site.webmanifest">



    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale-1, shrink-to-fit=no">
    <title>{% block title%}{% endblock %}</title>
    <link rel="stylesheet" type="text/css" href="static\styles\main.css">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></script>
  </head>
  <body>
    <div class="background">
      <div class="login_section">
        <img src="static\images\poll.jpg" class="image mobile-hide">
          <!--Flass message-->
          {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
              <div class="alert alert-warning" role="alert">
                {{ messages[0][1] }}
              </div>

            {% endif %}
          {% endwith %}

        {% block content %}

        {% endblock %}
      </div>
    </div>
  </body>
</html>
```

- **signin.html:**

```
{% from 'form_helper.html' import displayField %}

{% extends "prelogin-layout.html" %}

{% block title%}Registration {% endblock %}

{% block content %}
  <div class="form">
    <h2 class="heading">Get Started</h2>

    <form action="{{ url_for('signin') }}", method="POST">

      {{ displayField(form.username, 'Username', autocomplete="off", autofocus=true) }}
      {{ displayField(form.password, 'Password') }}
      {{ displayField(form.confirm_pswd, 'Confirm password') }}

      <div class="form-group">
        <input type="submit" value="Create" class="btn btn-secondary">
      </div>

      {{ form.csrf_token }}
    </form>

    <hr>

    <p class="bottom_msg">Already Registered? <a href={{url_for('login')}} class="link">Login here</a>.</p>
  </div>

{% endblock %}
```

- **chat.css:**

```
.show-mobile{
   display: none !important;
}

#hide{
   display: none;
}

.transparent-bar{
   display: none;
}

/* rooms name section */
#sidebar{
   position: absolute;
   margin-left: 85%;
   width: 15%;
   height: 100%;
   background-color: rgb(31, 30, 34);
   color: white;
}

#sidebar_rooms{
   padding-left: 2em;
   padding-right: 2em;
   padding-top: 1em;
   margin-top: 2em;
   height: 86%;
   width: 100%;
   color: white;
   overflow-y: auto; /*Make  this section scrolable */
}

/* that small verticle line at side of rooms name section */
.sidebar_with_tilte{
   position: absolute;
   background-color: rgb(131, 131, 131);
   width: 0.05em;
   height: 100%;
   margin-left: 0%;
}

/* "ROOMS" text */
.rooms_text{
   color: white;
   font-family: montserrat;
   text-align: center;
   padding-top :2rem;
}
```

```
/* each room name */
.select-room{
    font-size: medium;
}

/* each room text when hovered */
.select-room:hover{
    background-color: #3b3b3b7e;
    text-align: center !important;
    padding: 0.5em !important;
    border-radius: 20px !important;
    cursor: pointer !important;
}

.selected_img{
    display: none;
    position: absolute;
    background-color: black;
    color: white;
    font-family: montserrat;
    border-color: rgb(255, 255, 255);
    border-width: 1px;
    border-style: solid;
    margin-left: 7%;
    width: 70%;
    bottom: 4rem;
    min-height: 3em;
    border-radius: 25px;
}

#img_output{
    margin: 1em;
    width: 500px;
}

.fa-times-circle{
    position: absolute;
    top: 0.5em;
    right: 0.4em;
}

.fa-times-circle:hover{
    cursor: pointer;
}

.fa-file-image{
    position: absolute;
    bottom: 1.25em;
    color: rgb(0, 0, 0);
    right: 24%;
}
```

```
.fa-file-image:hover{
   cursor: pointer;
   color: rgb(97, 58, 58);
}

/* rounded input box */
.form-rounded {
   position: absolute;
   border-radius: 1rem !important;
   width: 70% !important;
   margin-left: 7%;
   bottom: 1rem;
   padding-right: 3em !important;
}

/* send button */
.btn-circle {
   background-color: rgb(6, 185, 176) !important;
   position: absolute;
   width: 50px;
   height: 50px;
   line-height: 45px;
   text-align: center;
   padding: 0;
   border-radius: 50% !important;
   bottom: 0.6rem;
   right: 18%;
  }

/* send button icon */
.fa-arrow-right{
   color: white;
}

/* whole section-dont know if its important */
#rightside-pannel {
   overflow: hidden; /* Make this div take up the rest of the horizontal space, and no more */
}

/* whole back section */
.main-chat-section{
   background: rgb(53,53,53);
   background: linear-gradient(114deg, rgba(53,53,53,1) 0%, rgba(11,11,11,1) 100%);
   width: 100%;
   height: 100%;
}

/* the area where msg is displayed */
#display-message-section {
   padding-top: 2em;
   min-height: 90%;
```

```
    max-height: 90%;
    overflow-y: auto; /*Make  this section scrolable */
}

.my-img{
    margin-left: 55%;
    max-width: 25em;
    background-color: #383838;
    color: white;
    font-family: montserrat;
    border-color: rgb(101, 102, 40);
    border-width: 1px;
    border-style: solid;
    word-wrap: break-word;
}

.other-img{
    margin-left: 8%;
    max-width: 25em;
    background-color: #383838;
    color: white;
    font-family: montserrat;
    border-color: rgb(52, 89, 114);
    border-width: 1px;
    border-style: solid;
    word-wrap: break-word;
}

.chat-img{
    width: 24.9em;
}

.enlarged-img{
    display: none;
    position: absolute;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.795);
    z-index: 2;
}

.open_img{
    max-height: 100%;
    max-width: 100%;
    width: auto;
    height: auto;
    position: absolute;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    margin: auto;
```

```
}

.fa-arrow-left{
    position: absolute;
    color: white;
    margin-top: 1em;
    margin-left: 1em;
}

.fa-cloud-download-alt{
    position: absolute;
    color: white;
    right: 1em;
    bottom: 1em;
}

.fa-arrow-left:hover{
    cursor: pointer;
}

.fa-cloud-download-alt{
    cursor: pointer;
}

.others-msg {
    margin-left: 8%;
    max-width: 25em;
    padding: 0.5em 0.5em 0.5em 1em;
    border-radius: 0px 15px 15px 15px;
    background-color: #242323;
    color: white;
    font-family: montserrat;
    border-color: rgb(52, 89, 114);
    border-width: 0.5px;
    border-style: solid;
    word-wrap: break-word;
    padding-right: 1em;
}

.my-msg {
    margin-left: 55%;
    max-width: 25em;
    padding: 0.5em 0.5em 0.5em 1em;
    border-radius: 15px 15px 0px 15px;
    background-color: #383838;
    color: white;
    font-family: montserrat;
    border-color: rgb(101, 102, 40);
    border-width: 0.5px;
    border-style: solid;
    word-wrap: break-word;
    padding-right: 1em;
```

```
}

.system-msg {
   color: black;
   font-size:0.9em;
   text-align: center;
   margin-left: 32%;
   max-width: 30%;
   background-color: white;
   border-radius: 10px;
}

.timestamp {
   color: #898d92;
   font-size:0.7em;
}

.other-username {
   font-weight: bold;
   font-size: 1.1em;
   font-family: montserrat;
   color: #00b7ff;
}

.my-username {
   font-weight: bold;
   font-size: 1.1em;
   font-family: montserrat;
   color: #ffc400;
}

/*custom scrollbar*/
#sidebar::-webkit-scrollbar-track
{
   -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.3);
   box-shadow: inset 0 0 6px rgba(0,0,0,0.3);
   background-color: #F5F5F5;
}

#sidebar::-webkit-scrollbar
{
   width: 5px;
   background-color: #F5F5F5;
}

#sidebar::-webkit-scrollbar-thumb
{
   background-color: rgb(2, 96, 204);
   background-image: -webkit-linear-gradient(45deg,
                                    rgba(255, 255, 255, .2) 25%,
                                    transparent 25%,
                                    transparent 50%,
```

```
                                rgba(255, 255, 255, .2) 50%,
                                rgba(255, 255, 255, .2) 75%,
                                transparent 75%,
                                transparent)
}

/* hiding scrollbar in chrome */
#display-message-section::-webkit-scrollbar {
   display: none;
 }

 /* Hide scrollbar for IE, Edge and Firefox */
#display-message-section {
   -ms-overflow-style: none;  /* IE and Edge */
   scrollbar-width: none;  /* Firefox */
 }

 input[type="file"] {
   display: none;
 }

@media only screen and (max-width: 875px){
   .show-mobile{
      display: block !important;
   }

   .mobile-hide{
      display: none;
   }

   .transparent-bar{
      background-color: rgba(0, 0, 0, 0.5);
      position: absolute;
      right: 0;
      width: 40%;
      height: 100%;
   }

   #display-message-section {
      padding-top: 4em;
   }

   #sidebar{
      position: absolute !important;
      margin-left: 13%;
      width: 50%;
      z-index: 2;
   }

   .my-img{
      right: 0;
      margin-right: 5%;
```

```
    max-width: 30em;
    margin-left: 15%;
}

.other-img{
    right: 0;
    margin-left: 5%;
    max-width: 30em;
    margin-right: 15%;
}

.my-msg{
    right: 0;
    margin-right: 5%;
    max-width: 30em;
    margin-left: 15%;
}

.chat-img{
    max-width:100%;
    height:auto;
}

.others-msg{
    margin-left: 5%;
    max-width: 30em;
    margin-right: 15%;
}

.system-msg{
    margin-left: 20%;
    max-width: 60%;
}

.selected_img{
    margin-left: 4%;
    width: 75%;
    bottom: 4em;
}

#img_output{
    width: 200px;
}

.form-rounded{
    margin-left: 4%!important;
    width: 75% !important;
    bottom: 1em;
}

.fa-file-image{
    position: absolute;
```

```
      bottom: 1.25em;
      right: 24%;
    }

    .btn-circle{
      right: 4%;
    }
}
```

- **create_room.css:**

```
@import url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,600;1,400&display=swap');

.background_create_room{
   margin: 0;
   background: rgb(53,53,53);
   background: linear-gradient(114deg, rgba(53,53,53,1) 0%, rgba(11,11,11,1) 100%);
   height: 100%;
   width: 100%;
   font-family: 'montserrat';
}

/* alert msgs */
.alert{
   position: absolute;
   margin-left: 15%;
   width: 74%;
   text-align: center;
}

/* center area */
.center_block{
   position: absolute;
   margin-left: 13%;
   margin-top: 5%;
   width: 80%;
   height: 80%;
   background-color: rgba(46, 46, 46, 0.85);
   border-radius: 15px 15px 15px 15px;
}

.create_room{
   color: #ffffff;
   position: absolute;
   margin-top: 5%;
   padding-top: 5%;
   margin-left: 10%;
   width: 30%;
   text-align: center;
}

.join_room{
   color: #ffffff;
   width: 30%;
   margin-left: 60%;
   margin-top: 5%;
   height: 80%;
   padding-top: 5%;
   text-align: center;
}
```

```
.text{
    font-size: 150%;
    margin-bottom: 4.5em;
}

.create_text{
    text-align: center;
}

.create_input{
    margin: 0 auto;
    display: block;
}

.btn-outline-light{
    height: 50px;
    width: 50%;
}

.create_button{
    margin: 0 auto;
    display: block !important;
    width: 30%;
}

.join_text{
    text-align: center;
}

.join_input{
    margin: 0 auto;
    display: block;
}

.join_button{
    margin: 0 auto;
    display: block !important;
    width: 30%;
}

@media only screen and (max-width: 875px){

    .background_create_room{
        height: 120%;
    }

    .center_block{
        margin-left: 7%;
        width: 86%;
        margin-top: 5em;
        padding-bottom: 1em;
```

```
    height: auto;
  }

  .alert{
    position: absolute;
    padding-top: 3em;
  }

  .create_room{
    position: relative;
    width: 100%;
  }

  .join_room{
    margin-top: 20%;
    position: relative;
    width: 100%;
    margin-left: 0;
    border-left: 0;
  }

  .create_room{
    margin-left: 5%;
    width: 90%;
  }

  .join_room{
    margin-left: 5%;
    width: 90%;
  }

}
```

- **header.css:**

```
body{
    margin:0;
}

.navbar_left{
    position: absolute;
    height: 98%;
    top: 2%;
    left: 1%;
    width: 3.5%;
}

.user{
    filter: invert(94%) sepia(0%) saturate(0%) hue-rotate(202deg) brightness(106%) contrast(105%);
    width: 60%;
    margin-left: 0.6rem;
    margin-top: 0.7%;
}

.user_name{
    size: 2em;
    color: #ffffff;
    text-align: center;
}

/* join room icon on the navbar */
.join_room_icon{
    filter: invert(94%) sepia(0%) saturate(0%) hue-rotate(202deg) brightness(106%) contrast(105%);
    width: 60%;
    margin-left: 0.7rem;
    margin-top: 5%;
}

.logout_icon{
    filter: invert(94%) sepia(0%) saturate(0%) hue-rotate(202deg) brightness(106%) contrast(105%);
    width: 60%;
    margin-left: 0.7rem;
    margin-top: 35em;
}

@media only screen and (max-width: 875px) {
    .navbar_left{
        position: fixed;
        display: block;
        left: 0;
        top: 0;
        height: 100%;
        width: 13%;
        background-color: black;
```

```
    z-index: 2;
  }

  .top_bar{
    display: block;
    position: fixed;
    z-index: 1;
    background-color: rgb(0, 0, 0);
    top: 0;
    width: 99.9%;
    height: 3em;
    left: 0;
  }

  .fa-bars{
    position: fixed;
    color: white;
    z-index: 2;
    margin-top: 0.9em;
    margin-left: 0.65em;
  }

  .fa-bars:hover{
    cursor: pointer;
  }

  .user{
    margin-top: 3.7em;
  }

  .logout_icon{
    margin-top: 50%;
  }

}
```

- **main.css:**

```
@import url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,600;1,400&display=swap');

body{
   margin:0;
}

.alert{
   position: absolute;
   margin-left: 60%;
   width: 30%;
   height: 8%;
   text-align: center;
}

.background{
   background: rgb(53,53,53);
   background: linear-gradient(114deg, rgba(53,53,53,1) 0%, rgba(11,11,11,1) 100%);
    height: 100%;
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
}

.login_section{
   margin-top: 4%;
   margin-left: 15%;
   background-color: rgb(255, 255, 255);
   height: 85.2%;
   width: 68%;
   border-radius: 25px 25px 25px 25px;
   box-shadow: 0 2px 20px 0 rgba(0, 0, 0, 0.13);
}

.image {
   position: absolute;
   border-radius: 25px 2px 2px 25px;
   width: 30%

}

.heading{
   font-family: 'montserrat', sans-serif;
   font-size: 300%;
   margin-bottom: 1em;
}

.form {
   position: absolute;
   margin-top: 5%;
```

```
    margin-left: 40%;


}

.formError {
    list-style-type:none;
    padding:0px;
    color: red;
}

.btn {
    margin-top: 30px;
    width: 120px !important;
    height: 50px;
    margin-left: 0.5em;
    background-color:rgb(226, 109, 80) !important;
    font-family: 'montserrat';


}

input[type='text'], input[type='password'] {
   margin-bottom: 2.5em;
   width: 20em;
}

.hr{
    margin-top: 3em;
    margin-bottom: 3em;
}

.bottom_msg{
    font-family: 'montserrat';
    font-size: small;
}

.link {
    color: black;
    text-decoration: underline;
}

@media only screen and (max-width: 875px){
    .mobile-hide{
        display: none;
    }

    .alert{
        margin: auto;
        display: block;
        width: 90%;
        padding: 0 auto;
```

```
    margin-top: 5%;
  }

  .login_section{
    margin-top: 20%;
    margin-left: 5%;
    height: 80%;
    width: 90%;
  }

  .form{
    position: relative;
    margin-left: 0%;
    top: 14%;
  }

  .heading{
    position: relative;
    text-align: center;
  }

  input[type='text'], input[type='password'] {
    margin: auto;
    display: block;
    margin-bottom: 2.5em;
    width: 85%;
  }

  .btn{
    margin: auto;
    display: block !important;
  }

  .bottom_msg{
    text-align: center;
  }
}
```

- **chat_page.js:**

```
document.addEventListener('DOMContentLoaded', () => {

  //ham button mobile version
  document.querySelector("#ham").onclick = () => {
    document.querySelectorAll(".mobile-hide").forEach(p => {
      p.style.display = "block";
    });
  }
  //ham exit
  document.querySelector("#ham_exit").onclick = () =>{
    document.querySelectorAll(".mobile-hide").forEach(p => {
      p.style.display="none";
    });

  }
  //the right portion of side menu, when clicked the side menu is closed
  document.querySelector(".transparent-bar").onclick = () =>{
    document.querySelectorAll(".mobile-hide").forEach(p => {
      p.style.display="none";
    });
  }
  //image icon when clicked an image preview is shown
  document.querySelector("#send_image").onclick = () =>{
    document.querySelector(".selected_img").style.display="block";
  }
  //when a file is selected it is shown in the preview
  const fileSelector = document.getElementById('img_upload');
  fileSelector.addEventListener('change', (event) => {
    const fileList = event.target.files;
    if (fileList[0]){
      //var img_name = fileList[0].name;
      //var img_size = fileList[0].size;
      // document.getElementById("image_name").innerHTML = "<ul><li>"+img_name+"    "+img_size+" "+"Bytes"+"</li></ul>";
      var image = document.getElementById('img_output');
      image.style.display = "block";
      image.src = URL.createObjectURL(event.target.files[0]);
    }else{
      var image = document.getElementById('img_output');
      image.style.display = "none";
      document.querySelector(".selected_img").style.display="none";
    }
  });

  //cancel image button
  document.querySelector('#cancel-img').onclick = () =>{
    document.getElementById('img_upload').value = "";
    var image = document.getElementById('img_output');
    image.style.display = "none";
    document.querySelector(".selected_img").style.display="none";
  }
```

```
  //download button
  document.querySelector('#download-enlarged-image').onclick = () =>{
    var img = document.querySelector("#open_img");
    var a = document.createElement("a"); //Create <a>
    a.href = img.src; //Image Base64 Goes here
    a.download = "Image.jpg"; //File name Here
    a.click(); //Downloaded file
  }
  //close button on image view
  document.querySelector('#close-enlarged-img').onclick = () =>{
    document.querySelector("#enlarged-img").style.display = "none";
  }

  // Make 'enter' key submit message
  let msg = document.querySelector("#user_message");
  msg.addEventListener("keyup", event => {
    event.preventDefault();
    if (event.keyCode === 13) {
      document.querySelector("#send_message").click();
    }
  });

});
```

- **socketio.js:**

```
document.addEventListener('DOMContentLoaded', () => {

  //connect ot socket.io
  var socket = io.connect(location.protocol + '//' + document.domain + ':' + location.port);

  //if there is a room_name (came here from create_room page) it joins the room and if not (came here from login page) makes the input bar invisible.
  room = room_name;
  if (room_name != "") {
    joinRoom(room);
  } else {
    document.getElementById("user_message").style.visibility="hidden";
    document.getElementById("send_message").style.visibility="hidden";
    document.getElementById("send_image").style.visibility="hidden";
    msg = "Select a room and start chating!";
    printSysMsg(msg);
  }

  // displays incomming messages
  // Display all incoming messages
  socket.on('message', data => {
    print_msg_other(data,username);
  });

  socket.on('img', img_data => {
    if (img_data['username'] != username){
      printOtherImg(img_data['img'],img_data['username']);
    }
  });

  // send message
  document.querySelector('#send_message').onclick = () =>{
    if (document.querySelector('#user_message').value != ""){
      var data = {'msg': document.querySelector('#user_message').value, 'username': username, 'room': room };
      print_my_msg(data,username);
      socket.send(data);
    }

    // converting the image into base24
    var imgSelected = document.getElementById("img_upload").files;
    if (imgSelected.length > 0){
      var fileToLoad = imgSelected[0];

      var fileReader = new FileReader();

      fileReader.onload = function(fileLoadedEvent) {
        var img_src = fileLoadedEvent.target.result;

        var img_data = {'img': img_src, 'username':username, 'room':room };
        printMyImg(img_src, username);  //print my sent image
```

```
            socket.emit('img', img_data);   // send the image to the server
        }
        fileReader.readAsDataURL(fileToLoad);
    }
    //when the send button the image preview is set to be hidden
    document.getElementById('img_upload').value = "";
    var image = document.getElementById('img_output');
    image.style.display = "none";
    document.querySelector(".selected_img").style.display="none";

    //clear input box
    document.querySelector('#user_message').value = '';
    document.querySelector('#user_message').focus();
};

//Room selesction
document.querySelectorAll('.select-room').forEach(p => {
    p.onclick = () => {
        //gets msg history from the databse(server) and displays when the user joins a new room
        socket.on('new_join_history', data => {
            socket.off('new_join_history');
            for (var i =0; i<data.length; i++){
                print_msg_history(data[i],username);
            }
        });

        // remove the menu
        if (window.innerWidth < 875) {
            document.querySelectorAll(".mobile-hide").forEach(p => {
                p.style.display="none";
            });
        }

        //making the input bar visible.
        document.getElementById("user_message").style.visibility="visible";
        document.getElementById("send_message").style.visibility="visible";
        document.getElementById("send_image").style.visibility="visible";
        let newRoom = p.innerHTML;
        if (newRoom != room) {
            leaveRoom(room);
            joinRoom(newRoom);
            room = newRoom;
        }
    };
});

//New room button
document.querySelector('#create_room').onclick = () => {
    leaveRoom(room_name)
}

//Logout button
```

```
document.querySelector('#logout_button').onclick = () => {
    leaveRoom(room_name)
}
//sends a logout message when the user refreshed or coses the tab
window.onbeforeunload = function () {
    socket.emit('leave', {'username': username, 'room': room});
}
//print other messags
function print_msg_other(data,username){
    // Display current message
    if (data.msg) {
        const p = document.createElement('p');
        const span_username = document.createElement('span');
        const span_timestamp = document.createElement('span');
        const br = document.createElement('br');
        // Display other users' messages
        if (data.username != username){
            if (typeof data.username !== 'undefined') {
                p.setAttribute("class", "others-msg");

                // Username
                span_username.setAttribute("class", "other-username");
                span_username.innerText = data.username;

                // Timestamp
                span_timestamp.setAttribute("class", "timestamp");
                span_timestamp.innerText = data.time_stamp;

                // HTML to append
                p.innerHTML += span_username.outerHTML + br.outerHTML + data.msg + br.outerHTML + span_timestamp.outerHTML;

                //Append
                document.querySelector('#display-message-section').append(p);
            }
            // Display system message
            else {
                // Checks if other user has joined or its our own msg.
                if (data.name != username){
                    printSysMsg(data.msg);
                }
            }
        }
    }
    scrollDownChatWindow();
}
//prints users message
function print_my_msg(data,username){
    // time
    var d = new Date();
    const monthNames = ["January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"];
    var m = d.getMonth();
```

```
    var date = d.getDate();
    var hr = d.getHours();
    var min = d.getMinutes();
    if (min < 10) {
        min = "0" + min;
    }
    var ampm = "AM";
    if( hr > 12 ) {
        hr -= 12;
        ampm = "PM";
    }

    // Display current message
    if (data.msg) {
        const p = document.createElement('p');
        const span_username = document.createElement('span');
        const span_timestamp = document.createElement('span');
        const br = document.createElement('br')
        // Display user's own message
        if (data.username == username) {
            p.setAttribute("class", "my-msg");

            // Username
            span_username.setAttribute("class", "my-username");
            span_username.innerText = data.username;

            // Timestamp
            span_timestamp.setAttribute("class", "timestamp");
            span_timestamp.innerText = monthNames[m] + "-" + date + " " + hr +":" + min + ampm;

            // HTML to append
            p.innerHTML += span_username.outerHTML + br.outerHTML + data.msg + br.outerHTML + span_timestamp.outerHTML

            //Append
            document.querySelector('#display-message-section').append(p);
        }
    }
    scrollDownChatWindow();
}
//print users message
function printMyImg(img_src,username){
    if (img_src) {
        const p = document.createElement('p');
        const span_username = document.createElement('span');
        const chat_img = document.createElement('img');
        const br = document.createElement('br')
        // Display user's own img
            p.setAttribute("class", "my-img");
            chat_img.setAttribute("class", "chat-img");
            chat_img.src = img_src;

            // Username
```

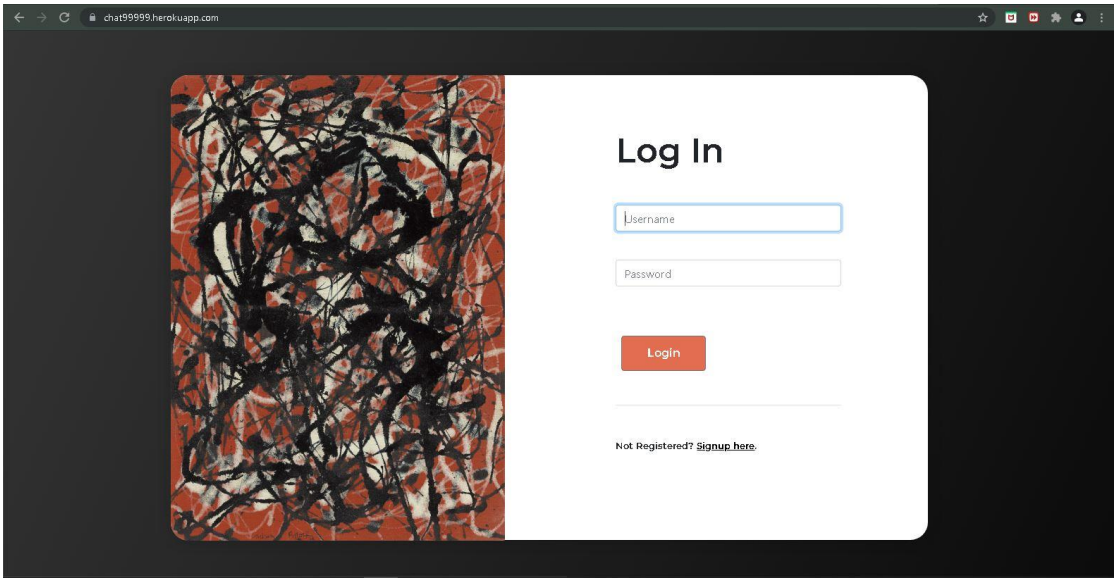
```
        span_username.setAttribute("class", "my-username");
        span_username.innerText = username;

        // HTML to append
        p.innerHTML += span_username.outerHTML + br.outerHTML + chat_img.outerHTML;

        //Append
        document.querySelector('#display-message-section').append(p);

        //open image
        document.querySelectorAll('.chat-img').forEach(p => {
          p.onclick = () => {
            document.querySelector(".enlarged-img").style.display = "block";
            var img_src = p.src;
            document.querySelector("#open_img").src = img_src;
          }
        });
    }
    scrollDownChatWindow();
}
//print others sent image
function printOtherImg(img_src,username){
    if (img_src) {
      const p = document.createElement('p');
      const span_username = document.createElement('span');
      const chat_img = document.createElement('img');
      const br = document.createElement('br')
      // Display other user's img
        p.setAttribute("class", "other-img");
        chat_img.setAttribute("class", "chat-img");
        chat_img.src = img_src;

        // Username
        span_username.setAttribute("class", "other-username");
        span_username.innerText = username;

        // HTML to append
        p.innerHTML += span_username.outerHTML + br.outerHTML + chat_img.outerHTML;

        //Append
        document.querySelector('#display-message-section').append(p);

        //open image
        document.querySelectorAll('.chat-img').forEach(p => {
          p.onclick = () => {
            document.querySelector(".enlarged-img").style.display = "block";
            var img_src = p.src;
            document.querySelector("#open_img").src = img_src;
          }
        });
    }
    scrollDownChatWindow();
```

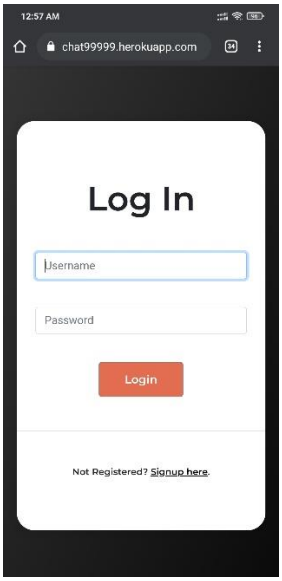
```
}
//print the message history
function print_msg_history(data,username){
   // Display current message
   if (data.msg) {
      const p = document.createElement('p');
      const span_username = document.createElement('span');
      const span_timestamp = document.createElement('span');
      const br = document.createElement('br')
      // Display user's own message
      if (data.username == username) {
            p.setAttribute("class", "my-msg");

            // Username
            span_username.setAttribute("class", "my-username");
            span_username.innerText = data.username;

            // Timestamp
            span_timestamp.setAttribute("class", "timestamp");
            span_timestamp.innerText = data.time_stamp;

            // HTML to append
            p.innerHTML += span_username.outerHTML + br.outerHTML + data.msg + br.outerHTML + span_timestamp.outerHTML

            //Append
            document.querySelector('#display-message-section').append(p);
      }
      // Display other users' messages
      else if (typeof data.username !== 'undefined') {
         p.setAttribute("class", "others-msg");

         // Username
         span_username.setAttribute("class", "other-username");
         span_username.innerText = data.username;

         // Timestamp
         span_timestamp.setAttribute("class", "timestamp");
         span_timestamp.innerText = data.time_stamp;

         // HTML to append
         p.innerHTML += span_username.outerHTML + br.outerHTML + data.msg + br.outerHTML + span_timestamp.outerHTML;

         //Append
         document.querySelector('#display-message-section').append(p);
      }
   }
   scrollDownChatWindow();
}

//Leave room
function leaveRoom(room){
   socket.emit('leave', {'username': username, 'room': room});
```

```
    //non-highlighting the room name
    document.querySelectorAll('.select-room').forEach(p => {
      p.style.backgroundColor = "";
      p.style.textAlign = "left";
      p.style.padding = "0";
      p.style.borderRadius = "0";
    });
  }

  //joinroom
  function joinRoom(room){
    socket.emit('join', {'username': username, 'room': room});

    // Highlight selected room
    document.querySelector('#' + CSS.escape(room)).style.backgroundColor = "#0704b1";
    document.querySelector('#' + CSS.escape(room)).style.textAlign = "center";
    document.querySelector('#' + CSS.escape(room)).style.padding = "0.5em";
    document.querySelector('#' + CSS.escape(room)).style.borderRadius = "20px";

    // Clear message area
    document.querySelector('#display-message-section').innerHTML = ''
    // autofocus on the textbox
    document.querySelector('#user_message').focus();

    //printing you joined the room
    printSysMsg("You joined the '" + room + "' room.")
  }

  // Scroll chat window down
  function scrollDownChatWindow() {
    const chatWindow = document.querySelector("#display-message-section");
    chatWindow.scrollTop = chatWindow.scrollHeight;
  }

  // Print system messages
  function printSysMsg(msg) {
    const p = document.createElement('p');
    p.setAttribute("class", "system-msg");
    p.innerHTML = msg;
    document.querySelector('#display-message-section').append(p);
    scrollDownChatWindow()

    // Autofocus on text box
    document.querySelector("#user_message").focus();
  }
});
```

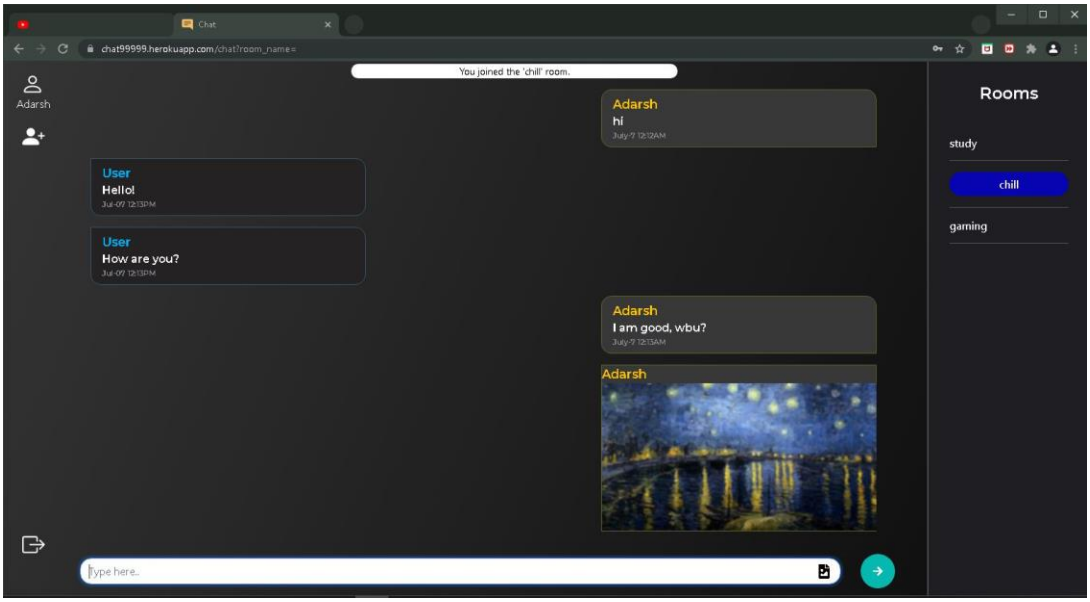
Output

The login page:

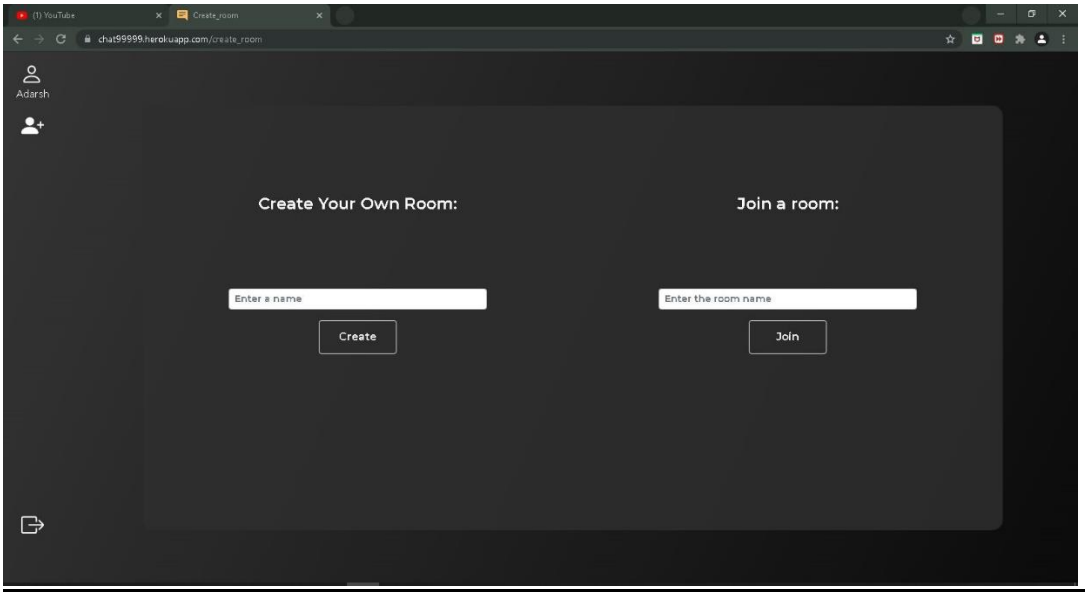


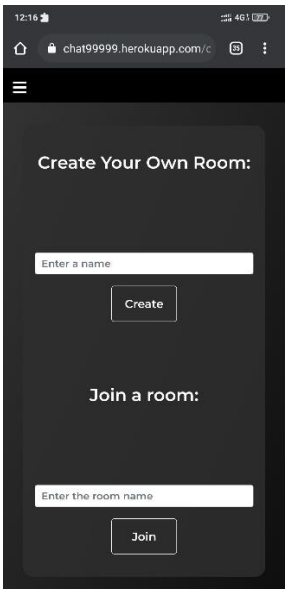


The Chat room page

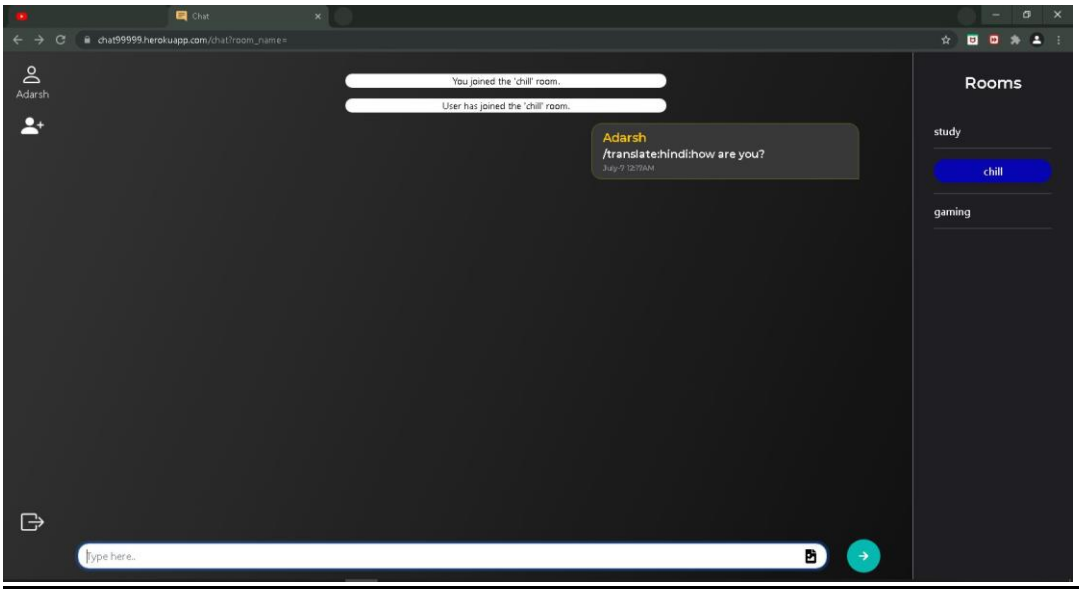


The Join room page





The Translation feature in action



Advantages

- Data Privacy:

Login Details of the user are always encrypted using the SHA-156 algorithm, and also the chat history is not saved so the user can text freely without bothering about an unauthorized person reading their chat history.

- Private rooms:

Users can create private chat rooms which they can then share with their friends and family and add them into the rooms and chat privately.

- Image Sharing:

Users can share images which helps them to share experiences that are not possible to share through words.

Disadvantage

- No message history:

Messages are not saved so once the user leaves the room all the previous messages are deleted. Message history can be added in the future but it can cause more frequent database bottlenecks.

- Database bottlenecking:

Because of the limited number of available connections to the database more than 10 users are not able to connect to the server. It can be solved by using more expensive plans from the server hosting.

Result and Discussion

Nowadays data has been a various serious concern, on Facebook, there are many allegations of data leak which somewhat extent is true, these companies are selling individual data to the companies so that those companies can use that data to know individuals interests and use it to sell their products to them. This is only a basic concern apart from that there are very serious cyber concerns are there.

So, to deal with these concerns this platform can provide a better alternative.

Conclusion

Chat99999 is an alternative to other chat applications, with its advantages and disadvantages. In this world, where data privacy is getting a bigger and bigger priority, this application can provide an alternative.

The project exposed us to the latest technology in the area of web development, network protocols, and server hosting.

Future Scope of Project

- Add the feature of storing messages so that users can see their past messages.
- Add the feature of voice calling and video calling down the line.
- Add the feature of sharing other files like videos, pdfs, text files, etc.
- Add the feature where users can play music in rooms.

References

- Flask-ScoketIO documentation: https://flask-socketio.readthedocs.io/en/latest/
- Flask documentation: https://flask.palletsprojects.com/en/2.0.x/
- Googletrans documentation: https://py-googletrans.readthedocs.io/en/latest/
- Bootstrap: https://getbootstrap.com/docs/4.1/getting-started/introduction/
- Fontawesome: https://fontawesome.com/
- Stackoverflow
- google