PyChat Project

running project

for sse connections redis is needed:

```
sudo apt install redis-server
```

create a virtual environment and install project's requirements using:

```
pip install -r requirements.txt
```

python version 3 should use!

create a local_params.py in root directory with following variables:

```
DB_HOST = 'db host'
DB_NAME = 'db name'
DB_USER = 'db user'
DB_PASS = 'db pass'

REDIS_URL = 'redis url'
MEM_CACHE_HOST = 'memcache url'

DEBUG_MODE = True or False
HOST = 'server running host'
```

mysql used for database.

create database using this SQL command:

```
CREATE SCHEMA `me_chat_test` DEFAULT CHARACTER SET utf8 ;
```

run the server using start.py

using project

first create a account.

login and use create a room or join room button get connected using room name.

in room creation first, enter room's name that you want then share the room's name with another user and waiting for another user to join. after that, using the start session button (red button) start the webrtc connection.

when you want to join a room, using the join a room button and enter the name of the room and join

project code explanation

the flask framework is used for writing the web application and for SEE connections, flask_sse used.

users send and receive its signals using a sse connection multiplexed by room name.

for example the user a want's a room with the user b.

firstly the user a creates a room with an arbitrary name and share it with the user b.

when the user a press the create button following function is called:

```
function createRoomBtn() {
        room_name = document.querySelector('#create_roomName').value;
        var stream_url = "{{ url_for('sse.stream') }}" + "?channel=" + room_name;
        sseConnection = new EventSource(stream_url);
        sseConnection.addEventListener('join', function (event) {
            var data = JSON.parse(event.data);
            if (data.username === "{{ user.username }}")
                notif("create", "you joined", "alert");
            else {
                notif("create", data.username + " joined", "alert");
                document.querySelector('#session_btn').disabled = false;
                document.querySelector('#createBtn').disabled = true;
                document.querySelector('#joinBtn').disabled = true;
            }

        }, null);
        sseConnection.addEventListener('answer', function (event) {
            json = JSON.parse(event.data);
            peerConnection.setRemoteDescription(new RTCSessionDescription(json.data));
            document.querySelector('#muteBtn').disabled = false;
            document.querySelector('#endBtn').disabled = false;
            document.querySelector('#offer').className = 'nav-item';
            document.querySelector('#answer').className = 'nav-item active';
        });
        sseConnection.addEventListener('candidate', function (event) {
            json = JSON.parse(event.data);
            peerConnection.addIceCandidate(json.data);
            document.querySelector('#answer').className = 'nav-item';
            document.querySelector('#established').className = 'nav-item active';
            if (timer == null)
                timer = setInterval(setTime, 1000);
        });
        document.querySelector('#create_notifications').innerHTML = '';
        var xhr = new XMLHttpRequest();
        xhr.open('POST', "{{ url_for('create_room') }}");
        xhr.onload = function () {
            if (this.status === 409) {
                notif("create", 'room already exists. try another one!', "alert");
            } else if (this.status === 400) {
                notif("create", 'room name cannot started with _user!', "alert");
            } else {
                notif("create", 'room created waiting for another people!', "alert");
                document.querySelector('#createBtn').disabled = true;
                document.querySelector('#joinBtn').disabled = true;
                document.querySelector('#createStatus').hidden = false;
                document.querySelector('#idle').className = 'nav-item';
                document.querySelector('#create').className = 'nav-item active';
            }

        };
        xhr.send(JSON.stringify({username: "{{ user.username }}", room: room_name}));
    }
```

in this function, the room name sends to the server, create an SSE connection that listens for the answer and joins messages. the answer message is used for webrtc connection and join is used to notifying the user when another user joined.

every room has its owned channel in SSE stream and room's user get messages using this channel.

using xhr this function sends a join message to other members.

the view that handles join requests is:

```
@App.route('/join_room', methods=['POST'])
@login_required
def join_room():
    data = loads(request.data)
    sse.publish({'username': data.get('username')}, type='join',
channel=data.get('room'))
    return Response('ok', status=200)
```

this function gives a username and publishes to other users using sse.publish().

other views that write for sending another kind of messages are like the mentioned view.

now the user b gives the room name and join to it.

when the user enters the room name and click the join button following function is called.

```
function joinRoomBtn() {
        console.log('join room called');
        room_name = document.querySelector('#join_roomName').value;
        var stream_url = "{{ url_for('sse.stream') }}" + "?channel=" + room_name;
        sseConnection = new EventSource(stream_url);
        document.querySelector('#joinStatus').hidden = false;
        document.querySelector('#idle2').className = 'nav-item';
        document.querySelector('#join').className = 'nav-item active';
        sseConnection.addEventListener('join', function (event) {
            var data = JSON.parse(event.data);
            if (data.username === "{{ user.username }}") {
                notif("join", "you joined", "alert");
            } else {
                notif("join", data.username + " joined", "alert");
            }
            document.querySelector('#join').className = 'nav-item';
            document.querySelector('#start2').className = 'nav-item active';

        }, null);

        sseConnection.addEventListener('offer', function (event) {
            json = JSON.parse(event.data);
            peerConnection.setRemoteDescription(new RTCSessionDescription(json.data));
            peerConnection.createAnswer().then(function (answer) {
                peerConnection.setLocalDescription(answer);
                var xhr = new XMLHttpRequest();
                xhr.open('POST', '{{ url_for('send_room_signal') }}');
                xhr.onload = function () {
                    if (!this.status === 200)
                        alert('failed to send answer');
                    else {
                        $('#joinModal').modal('hide');
                        document.querySelector('#muteBtn').disabled = false;
                        document.querySelector('#endBtn').disabled = false;
                        document.querySelector('#joinStatus').hidden = false;
                        document.querySelector('#start2').className = 'nav-item';
                        document.querySelector('#roffer').className = 'nav-item active';
                        document.querySelector('#join').className = 'nav-item';
                    }
                };
                xhr.send(JSON.stringify({data: answer, room: room_name, type:
'answer'}));
            });
        });
        sseConnection.addEventListener('candidate', function (event) {
            json = JSON.parse(event.data);
            peerConnection.addIceCandidate(json.data);
            document.querySelector('#roffer').className = 'nav-item';
            document.querySelector('#established2').className = 'nav-item active';
            if (timer == null) {
                console.log('timer');
                timer = setInterval(setTime, 1000);
            }

        });
        document.querySelector('#join_notifications').innerHTML = '';
        var xhr = new XMLHttpRequest();
        xhr.open('POST', "{{ url_for('join_room') }}");
        xhr.onload = function () {
            if (this.status === 200) {
                json = JSON.parse(this.responseText);
                json.members.forEach(function (entry) {
                    if (!entry === "{{ user.username }}")
                        notif("join", entry + " joined", "alert");
                });
                notif("join", json.creator + " joined", "alert");
                document.querySelector('#createBtn').disabled = true;
                document.querySelector('#joinBtn').disabled = true;
            } else if (this.status === 404)
                notif("join", 'room not found', "alert");
            else
                notif("join", 'error happened try again', "alert");
        };
        xhr.send(JSON.stringify({username: "{{ user.username }}", room: room_name}));
    }
```

in this function, SSE and RTC connection is created. in addition to joining messages this user receives the offer messages and after set rtc's remote description sends an answer to offerer user and connection is established between two users.

also, user state his presence in the room using the joinAnnouncement function that already mentioned.

when all users joined the room the room creator press start session button (that's now actives) and using sse all signals send and receives.

when the user presses the start session button following function is called:

```
function startSession(room_name) {
        notif("create", "session started", "alert");
        document.querySelector('#create').className = 'nav-item';
        document.querySelector('#start').className = 'nav-item active';
        peerConnection.createOffer().then(function (offer) {
            var xhr = new XMLHttpRequest();
            xhr.open('POST', '{{ url_for('send_room_signal') }}');
            xhr.onload = function () {
                if (!this.status === 200) {
                    alert('failed to send offer');
                } else {
                    $('#createModal').modal('hide');
                    document.querySelector('#start').className = 'nav-item';
                    document.querySelector('#offer').className = 'nav-item active';
                }

            };
            xhr.send(JSON.stringify({data: offer, room: room_name, type: 'offer'}));
            peerConnection.setLocalDescription(offer);
        }, function (error) {
            alert(error);
        })

    }
```

the start session function sends an offer to another user and its connection is established finally when all messages exchanged.

Web socket chat

using flask socket io both users can send texts to each other

code explanation

first a web socket object is created using this

```
socket_io = SocketIO(App)
```

then the views are register for it

```
@socket_io.on('event', namespace='/chat')
def test_msg(message):
    emit('response', message, namespace='/chat')


@socket_io.on('join', namespace='/chat')
@login_required
def on_join(data):
    data = loads(data)
    room_name = generate_room_name(current_user.username, data.get('username'))
    join_room(room_name)


@socket_io.on('send_message', namespace='/chat')
@login_required
def send_msg(data):
    json = loads(data)
    room_name = generate_room_name(json.get('sender'), json.get('receiver'))
    date = json.get('date')
    time = json.get('time')[0:8]
    datetime_obj = datetime.strptime('%s %s' % (time, date), '%H:%M:%S %a %b %d %Y')
    Message.create(
        sender=User.select().where(User.username == json.get('sender')).first().id,
        receiver=User.select().where(User.username == json.get('receiver')).first().id,
        msg=json.get('msg'), datetime=datetime_obj
    )
    json['datetime'] = str(datetime_obj)
    emit('message', dumps(json), room=room_name)


@App.route('/msgs/', methods=['POST'])
@login_required
def get_msgs():
    json = loads(request.data)
    user = User.select().where(User.username == json.get('username')).first().id
    msgs = Message.select().where(
        (Message.sender == current_user.id and Message.receiver == user) |
        (Message.receiver == current_user.id and Message.sender == user)
    ).order_by(Message.datetime)
    msgs_list = map(serialize_msg, msgs)
    return jsonify(list(msgs_list))
```

Creators

- Amin Hosseini
- Ali Mami Zade
- Amir Mahdi MirFakhar