Simplified version of an online HelpDesk

Project for course: Computer networks laboratory

Course code:15Z510

Team Members

Mani Sankar T	CSE	18Z229
Adwin Sanjo J	CSE	18Z202
Sri Nagul	CSE	18Z256
Arun Prashath	CSE	18Z208
Rohith	CSE	18Z242

PROBLEM STATEMENT

Simplified implementation of an online HelpDesk System.

INTRODUCTION

Online HelpDesk System nowadays are much faster and efficeint. Most of them are equiped with automated replies, but we humans still prefer a person over a machine.

Online helpDesk systems can be implemented via Websockets whichfacilitates the communication between the client and the server whithout the need to refresh the page, i.e. real time application.

- So how different is it from a regular chat Application?

  Here, we have multiple clients and multiple server side users.

  When a client sends a request for a query session, the program checks for availability of server side useers.These users are assigned randomly based on their availability at that particular time. If all users are occupied with the request of the other clients, it will just display that its not possible to assign a user to that client and asks the client to try after a while.

PACKAGES USED

SOCKETIO – Handles all socket connections

FLASK – light web application Framework; handles the backend

SQLAlchemy – converts to RAW SQL commands

HTML ,CSS , JS – Front End

PostgreSQL -Database

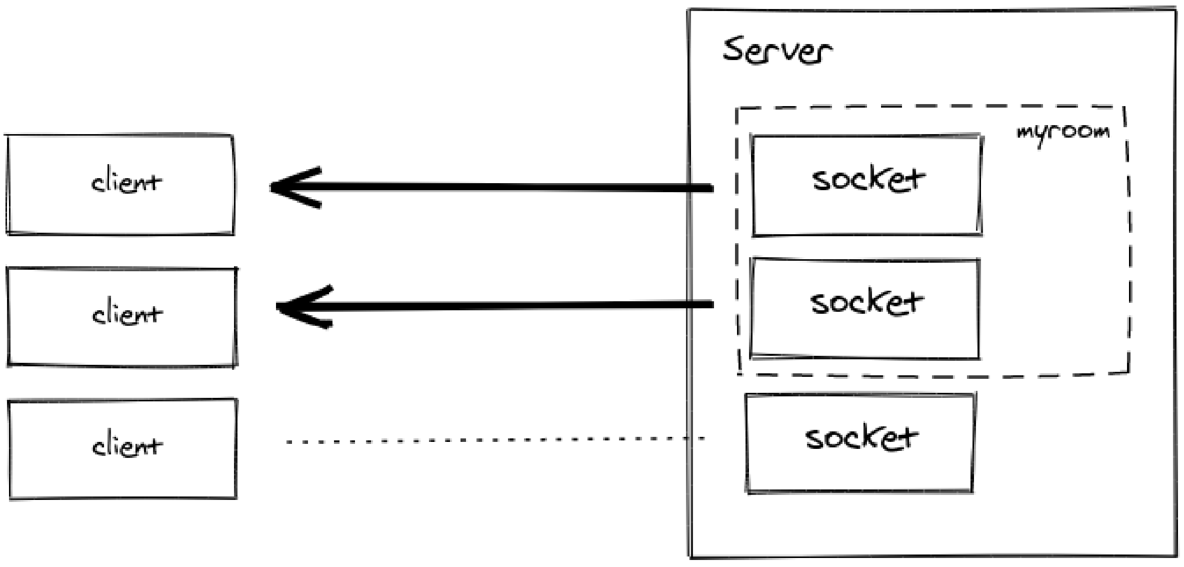
FUNCTIONALITIES

The functionalities provided in the actual application is as follows:

- Multiple server accounts/Instances
- Multiple customer
- Private chat based on Room concept in SocketIO
- Concept of Availability is invoked
- Ignore request functionality available for the users from the server side

ADVANTAGES

- No logging of data
- Close to instant pairing
- Not much user data is collected
- Can be extended to Discussion forums that can hold upto 250



Flask SQLAlchemy

socket.io

HTML JS CSS

Folder Structure

```
HELPDESK/
    - __pychache__/
    - static/
    - templates/
        -index.html
        -chat.html
        -server.html
        -serverchat.html
    - app.py
    - Pipfile
    - Pipfile.lock
    - config.cfg
```

Functions used in the program

```
from flask import Flask, render_template,flash
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import func
from flask_socketio import SocketIO,emit, join_room, leave_room
from flask import session, redirect, url_for, render_template, request
```

Socketio

```
socketio = SocketIO(app)
if __name__ == '__main__':
 socketio.run(app)
```

Database

(Hidden Sensitive data-username,password,host)

```
db = SQLAlchemy(app)
app.config.from_pyfile('config.cfg')
```

To join room

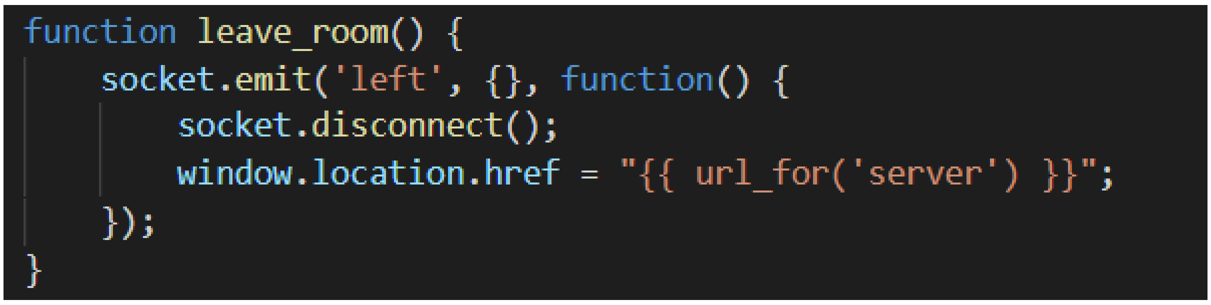
```
@socketio.on('joined', namespace='/chat')
def joined(message):
    room = session.get('room')
    join_room(room)
    emit('status', {'msg': 'Chat Active!'}, room=room)
```

To leave room

```
@socketio.on('left', namespace='/chat')
def left(message):
    room = session.get('room')
    leave_room(room)
```

```
  if(HELPDESK.query.filter_by(availability=room).first()) :
   helpDesk = HELPDESK.query.filter_by(availability=room).first()
   helpDesk.availability=0
   db.session.commit()
  emit('status', {'msg':'Terminated Chat'}, room=room)
```

Js in Serverchat.html -similar function in chat.html



```
function leave_room() {
    socket.emit('left', {}, function() {
        socket.disconnect();
        window.location.href = "{{ url_for('server') }}";
    });
}
```

To assign the server with the specific room to chat with the client

```
 @app.route('/serverchat/<NAME>/<ROOM>')

def serverchat(NAME,ROOM):
   session['name'] = NAME
   session['room'] = ROOM
   return redirect(url_for('schatting'))

@app.route('/schatting')
def schatting():
  name = session.get('name', '')
  room = session.get('room', '')
  if name == '' or room == '':
    return redirect(url_for('index'))
  return render_template('serverchat.html', name=name, room=ro
om)
```

To assign the client with the server in the same room

```
@app.route('/chat/<NAME>/<ROOM>')
def chat(NAME,ROOM):
   session['name'] = NAME
```

```
    session['room'] = ROOM
    if session['name'] == '' or session['room'] == '':
        return redirect(url_for('index'))
    helpDesk = HELPDESK.query.filter_by(availability=0).order_by(fun
c.random()).all()
    if(HELPDESK.query.filter_by(availability=0).order_by(func.random
()).all()):
        for x in helpDesk:
          x.availability=ROOM
          db.session.commit()
          break
        return redirect(url_for('chatting'))

    else :
       flash("Could not assign Anyone. Please try after some time.")
       return redirect(url_for('index'))
```

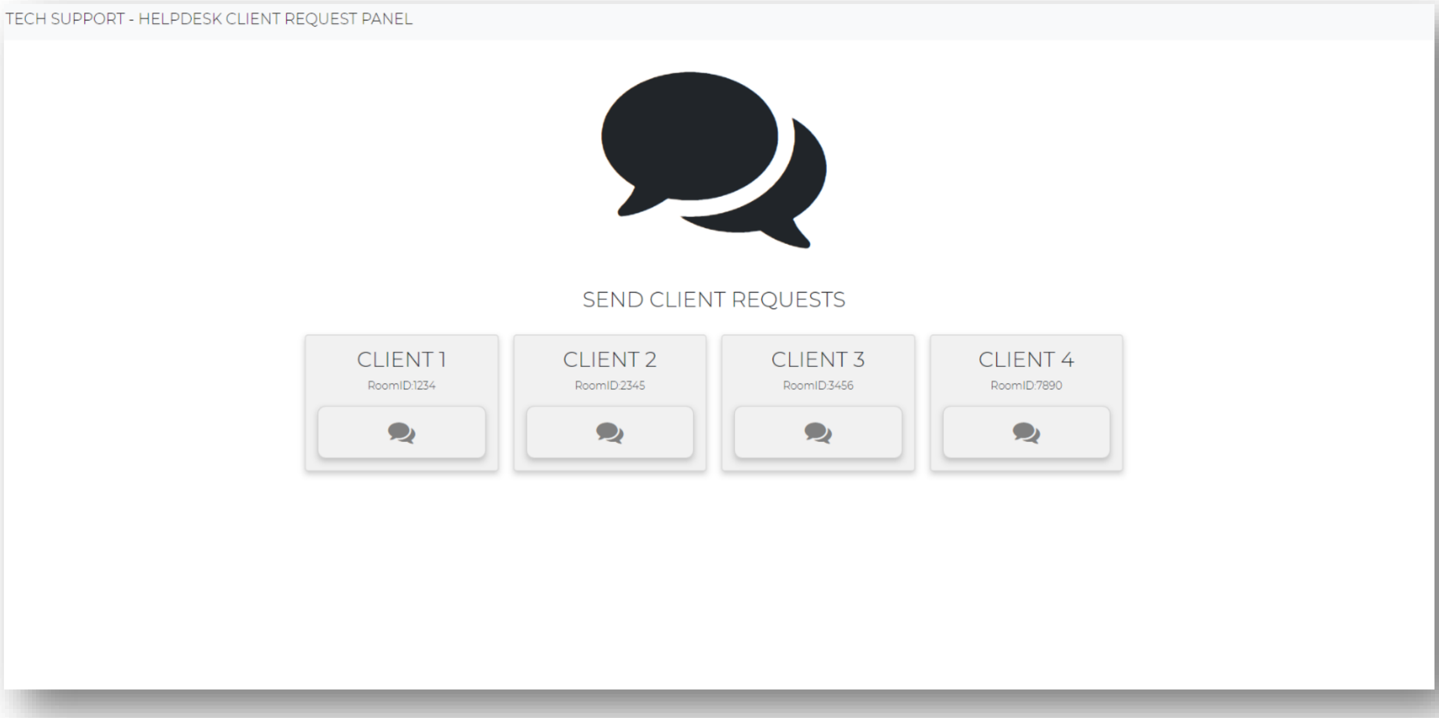
To transfer the message to the desired room

```
@socketio.on('text', namespace='/chat')
def text(message):
    room = session.get('room')
    emit('message', {'msg': message['msg']}, room=room)
```

The connection to a server is established by calling
the connect() method:

```
socket = io.connect('http://' + document.domain+':'+location.port
+'/chat);
                socket.on('connect', function() {
                    socket.emit('joined', {});
                });
```

Screenshots:


TECH SUPPORT - HELPDESK CLIENT REQUEST PANEL

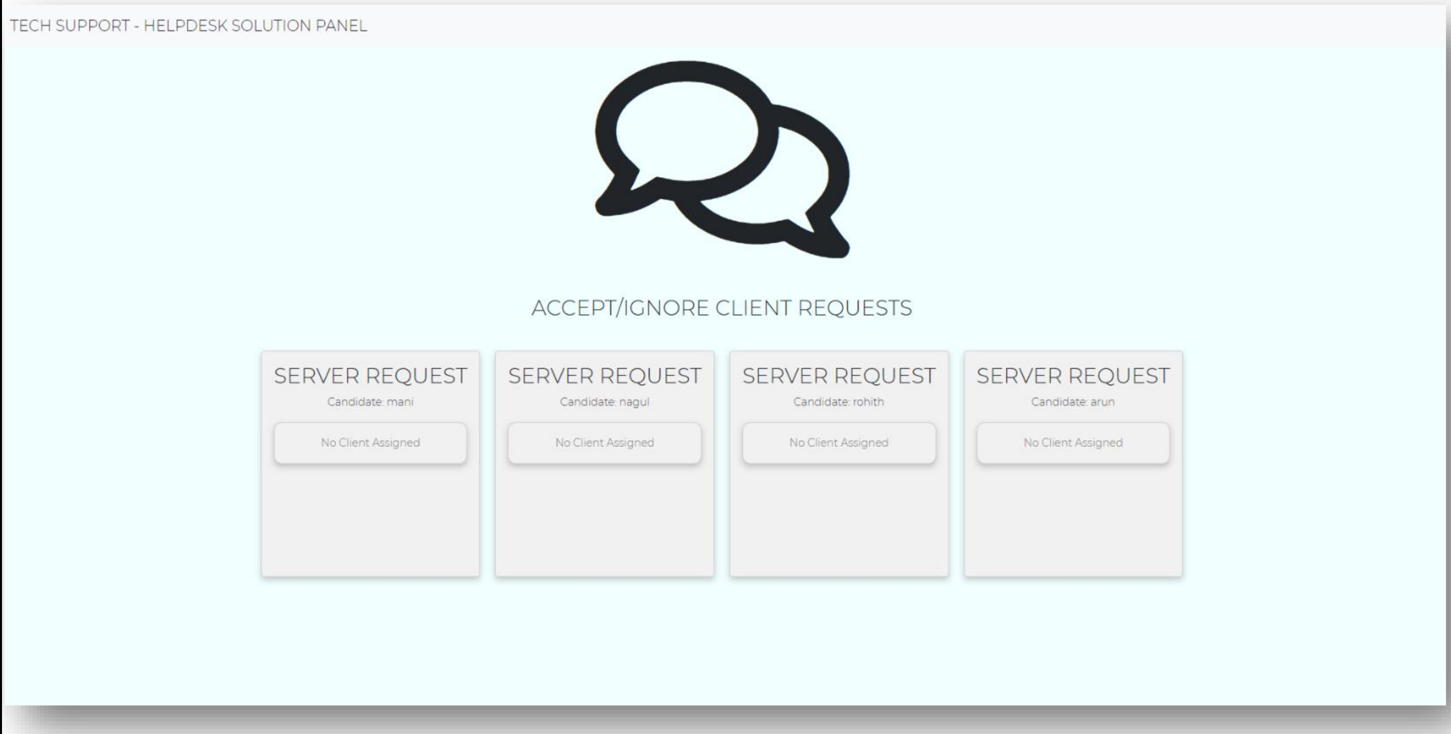
SEND CLIENT REQUESTS

CLIENT 1
RoomID:1234

CLIENT 2
RoomID:2345

CLIENT 3
RoomID:3456

CLIENT 4
RoomID:7890


TECH SUPPORT - HELPDESK SOLUTION PANEL

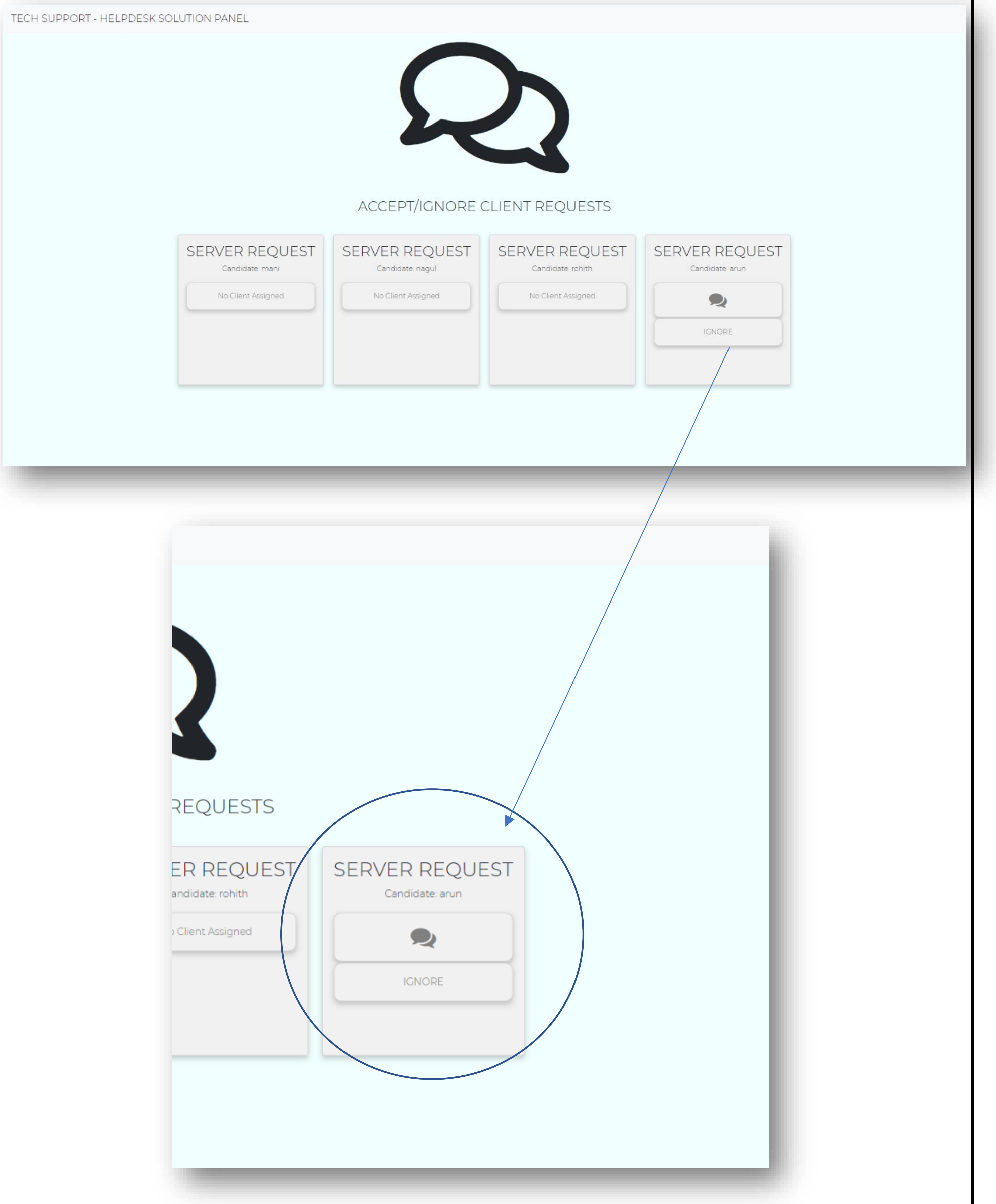
ACCEPT/IGNORE CLIENT REQUESTS

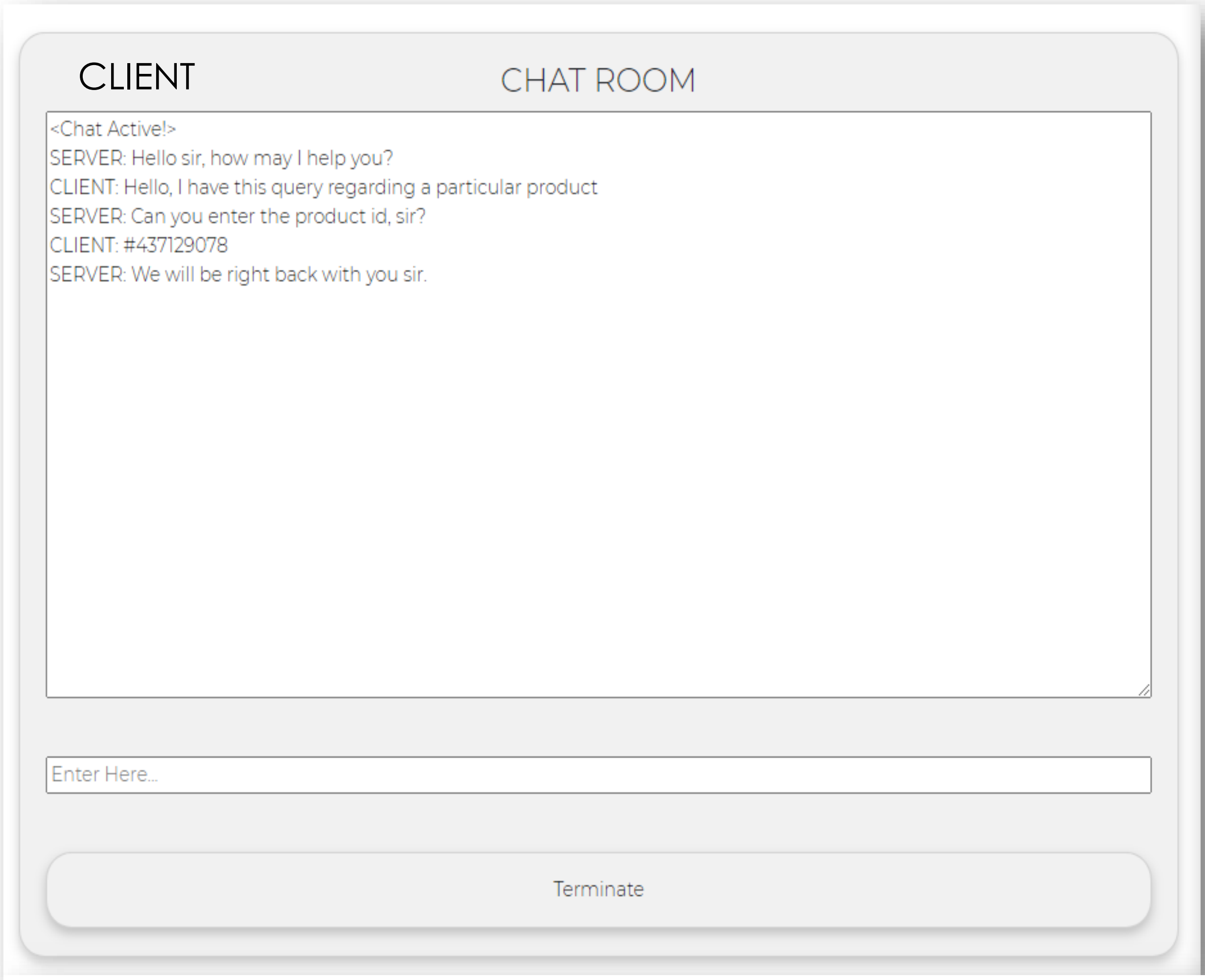
SERVER REQUEST
Candidate: mani
No Client Assigned

SERVER REQUEST
Candidate: nagul
No Client Assigned

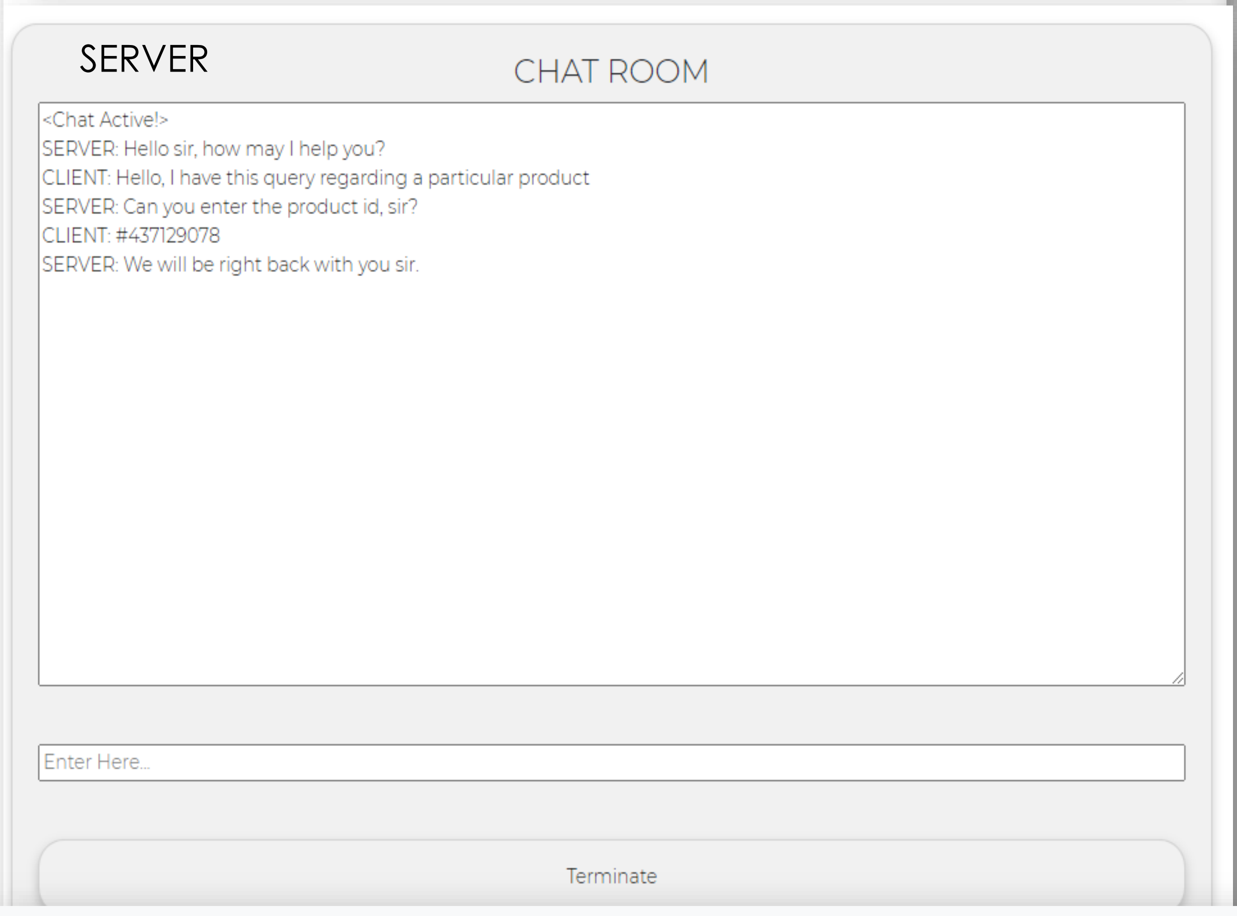
SERVER REQUEST
Candidate: rohith
No Client Assigned

SERVER REQUEST
Candidate: arun
No Client Assigned

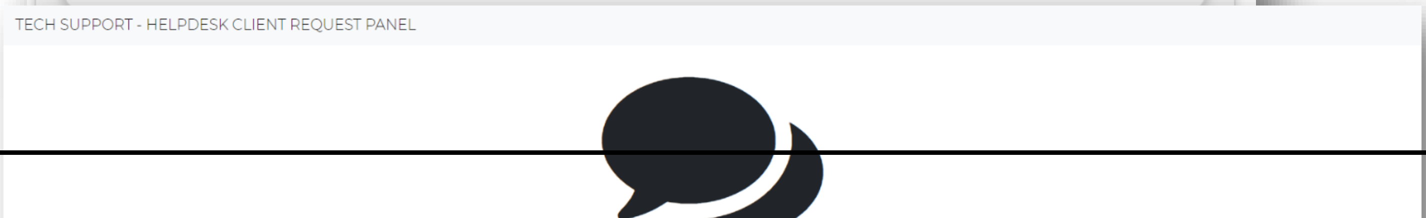

TECH SUPPORT - HELPDESK SOLUTION PANEL
ACCEPT/IGNORE CLIENT REQUESTS
SERVER REQUEST
Candidate: mani
No Client Assigned
SERVER REQUEST
Candidate: nagul
No Client Assigned
SERVER REQUEST
Candidate: rohith
No Client Assigned
SERVER REQUEST
Candidate: arun
IGNORE
REQUESTS
ER REQUEST
andidate: rohith
Client Assigned
SERVER REQUEST
Candidate: arun
IGNORE

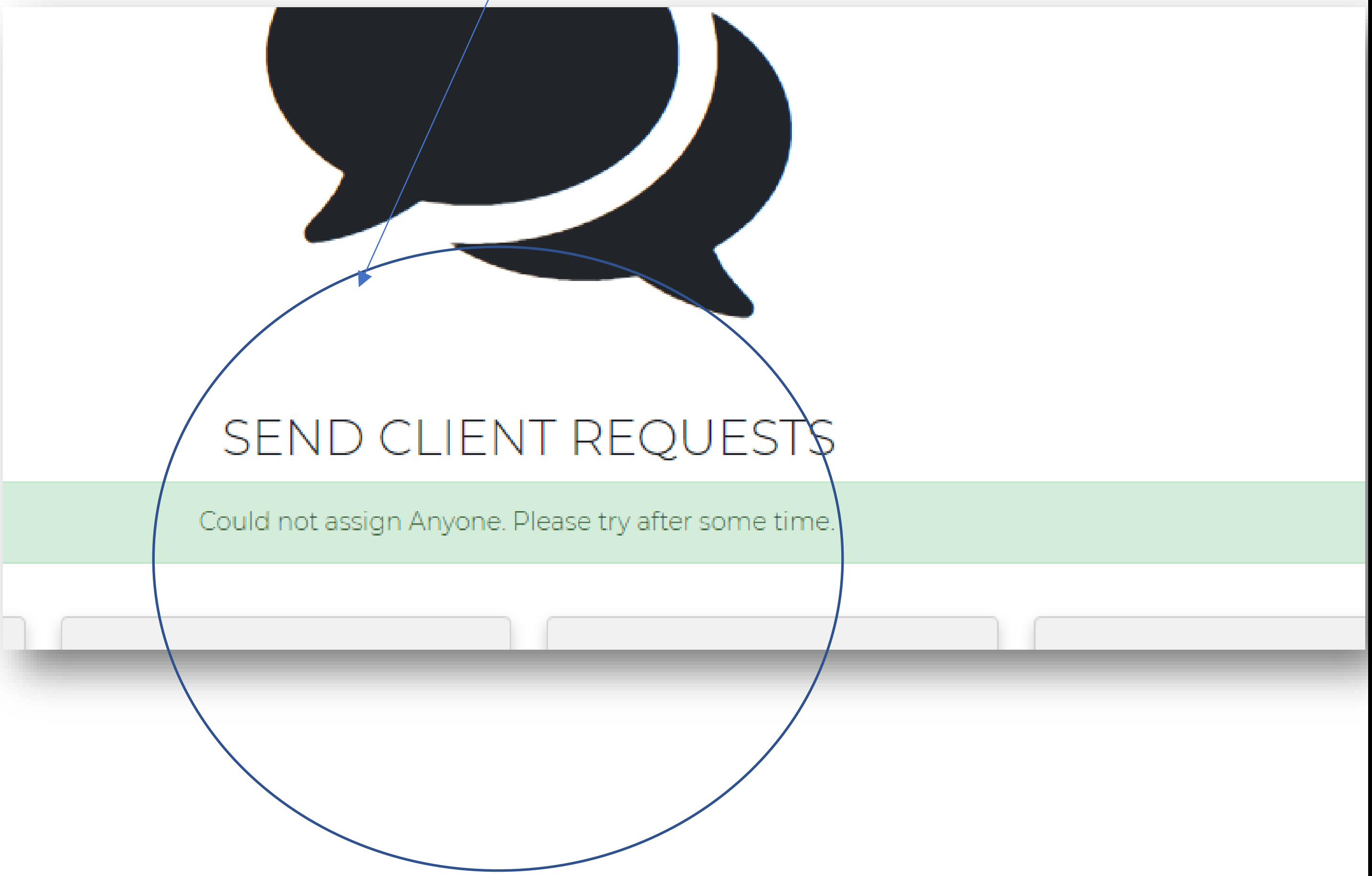


```
<Chat Active!>
SERVER: Hello sir, how may I help you?
CLIENT: Hello, I have this query regarding a particular product
SERVER: Can you enter the product id, sir?
CLIENT: #437129078
SERVER: We will be right back with you sir.
```



```
<Chat Active!>
SERVER: Hello sir, how may I help you?
CLIENT: Hello, I have this query regarding a particular product
SERVER: Can you enter the product id, sir?
CLIENT: #437129078
SERVER: We will be right back with you sir.
```





SEND CLIENT REQUESTS

Could not assign Anyone. Please try after some time.

Source code:

app.py:

```
from flask import Flask, render_template,flash
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import func
from flask_socketio import SocketIO,emit, join_room, leave_room
from flask import session, redirect, url_for, render_template, request


app= Flask(__name__)
app.config.from_pyfile('config.cfg')
db = SQLAlchemy(app)
socketio = SocketIO(app)
class HELPDESK(db.Model):
    __tablename__="Helpdesk"
    id = db.Column(db.String,primary_key=True)
    name = db.Column(db.String)
    availability = db.Column(db.Integer)
    def __init__(self,id,name,availability):
        self.id = id
        self.name = name
        self.availability = availability



#client
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/ignore/<id>')
def ignore(id):
    helpDesk = HELPDESK.query.get(id)
    print(helpDesk.name)
    room=helpDesk.availability
    helpDesk.availability=0
    db.session.commit()

    return redirect(url_for('server'))

@app.route('/chat/<NAME>/<ROOM>')
def chat(NAME,ROOM):
    session['name'] = NAME
    session['room'] = ROOM
    if session['name'] == '' or session['room'] == '':
        return redirect(url_for('index'))
    helpDesk = HELPDESK.query.filter_by(availability=0).all()
    if(HELPDESK.query.filter_by(availability=0).all()):
        for x in helpDesk:
          x.availability=ROOM
          db.session.commit()
          break
        return redirect(url_for('chatting'))

    else :
      flash("Sorry for the Inconvinience")
      return redirect(url_for('index'))

#server
@app.route('/server')
def server():
    HelpDesk = HELPDESK.query.all()
    return render_template('server.html',serv=HelpDesk)

@socketio.on('joined', namespace='/chat')
def joined(message):
    room = session.get('room')
    join_room(room)
    emit('status', {'msg': 'Chat Active!'}, room=room)


@socketio.on('text', namespace='/chat')
def text(message):
    room = session.get('room')
    emit('message', {'msg': message['msg']}, room=room)


@socketio.on('left', namespace='/chat')
def left(message):
    room = session.get('room')
    leave_room(room)
    if(HELPDESK.query.filter_by(availability=room).first()) :
     helpDesk = HELPDESK.query.filter_by(availability=room).first()
     helpDesk.availability=0
     db.session.commit()
    emit('status', {'msg':'Terminated Chat'}, room=room)
```

```
@app.route('/serverchat/<NAME>/<ROOM>')
def serverchat(NAME,ROOM):
    session['name'] = NAME
    session['room'] = ROOM
    return redirect(url_for('schatting'))

@app.route('/chatting')
def chatting():
  name = session.get('name', '')
  room = session.get('room', '')
  if name == '' or room == '':
    return redirect(url_for('index'))
  return render_template('chat.html', name=name, room=room)


@app.route('/schatting')
def schatting():
  name = session.get('name', '')
  room = session.get('room', '')
  if name == '' or room == '':
    return redirect(url_for('index'))
  return render_template('serverchat.html', name=name, room=ro
om)

if __name__ == '__main__':
    socketio.run(app,debug=True)
```

chat.html

```
from flask import Flask, render_template,flash
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import func
from flask_socketio import SocketIO,emit, join_room, leave_room
from flask import session, redirect, url_for, render_template, request


app= Flask(__name__)
app.config.from_pyfile('config.cfg')
db = SQLAlchemy(app)
```

```
socketio = SocketIO(app)
class HELPDESK(db.Model):
    __tablename__="Helpdesk"
    id = db.Column(db.String,primary_key=True)
    name = db.Column(db.String)
    availability = db.Column(db.Integer)
    def __init__(self,id,name,availability):
        self.id = id
        self.name = name
        self.availability = availability



#client
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/ignore/<id>')
def ignore(id):
    helpDesk = HELPDESK.query.get(id)
    print(helpDesk.name)
    room=helpDesk.availability
    helpDesk.availability=0
    db.session.commit()

    return redirect(url_for('server'))

@app.route('/chat/<NAME>/<ROOM>')
def chat(NAME,ROOM):
    session['name'] = NAME
    session['room'] = ROOM
    if session['name'] == '' or session['room'] == '':
        return redirect(url_for('index'))
    helpDesk = HELPDESK.query.filter_by(availability=0).order_by(fun
c.random()).all()
    if(HELPDESK.query.filter_by(availability=0).order_by(func.random
()).all()):
        for x in helpDesk:
          x.availability=ROOM
          db.session.commit()
          break
        return redirect(url_for('chatting'))

    else :
       flash("Sorry for the Inconvinience")
       return redirect(url_for('index'))

#server
@app.route('/server')
def server():
    HelpDesk = HELPDESK.query.all()
    return render_template('server.html',serv=HelpDesk)

@socketio.on('joined', namespace='/chat')
def joined(message):
    room = session.get('room')
    join_room(room)
    emit('status', {'msg': 'Chat Active!'}, room=room)


@socketio.on('text', namespace='/chat')
def text(message):
    room = session.get('room')
    emit('message', {'msg': message['msg']}, room=room)


@socketio.on('left', namespace='/chat')
def left(message):
    room = session.get('room')
    leave_room(room)
    if(HELPDESK.query.filter_by(availability=room).first()) :
     helpDesk = HELPDESK.query.filter_by(availability=room).first()
     helpDesk.availability=0
     db.session.commit()
    emit('status', {'msg':'Terminated Chat'}, room=room)


@app.route('/serverchat/<NAME>/<ROOM>')
def serverchat(NAME,ROOM):
    session['name'] = NAME
    session['room'] = ROOM
    return redirect(url_for('schatting'))
```

```
@app.route('/chatting')
def chatting():
  name = session.get('name', '')
  room = session.get('room', '')
  if name == '' or room == '':
    return redirect(url_for('index'))
  return render_template('chat.html', name=name, room=room)


@app.route('/schatting')
def schatting():
  name = session.get('name', '')
  room = session.get('room', '')
  if name == '' or room == '':
    return redirect(url_for('index'))
  return render_template('serverchat.html', name=name, room=ro
om)

if __name__ == '__main__':
    socketio.run(app,debug=True)
```

index.html

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/b
ootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1
/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3
Og8ifwB6" crossorigin="anonymous"></script>
  <link href="https://fonts.googleapis.com/css2?family=Montserrat:
wght@300&display=swap" rel="stylesheet">

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/f
ont-awesome/4.7.0/css/font-awesome.min.css">
<meta name="viewport" content="width=device-width, initial-
scale=1">
```

```
<style>
* {
  box-sizing: border-box;
}

body {font-family: 'Montserrat', sans-serif;font-size:14px;
}

.column {
  float: left;
  width: 25%;
  max-width:300px;
  padding: 0 10px;
}

.row {margin: 0 -5px;}

.row:after {
  content: "";
  display: table;
  clear: both;
}
@media screen and (max-width: 600px) {
  .column {
    width: 100%;
    display: block;
    margin-bottom: 20px;
  }
}

.card {
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
  padding: 16px;
  text-align: center;
  background-color: #f1f1f1;
}
</style>
</head>
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">
```

```
     <div class="display-
7">TECH SUPPORT - HELPDESK CLIENT REQUEST PANEL</div>
  </a>
</nav>
<body style="text-align: center;">
  <i class="fa fa-comments" style="font-size:300px"></i>
  <h3 style="text-align: center;padding-
top: 30px;;">SEND CLIENT REQUESTS</h3>

  {% with messages = get_flashed_messages() %}
    {% if messages %}
    {% for message in messages %}
    <div id="alertmessage" class="alert alert-success alert-
dismissable" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-
label="close">
            <span aria-
hidden="true" onclick="CLOSEFunction()">&times;</span>
        </button>
    {{message}}
  </div>
    {% endfor %}
    {% endif %}
    {% endwith %}
<script>
  function CLOSEFunction(){
    document.getElementById('alertmessage').style.display="none";
  }
</script>
<div class="row container" style="margin-top:30px;justify-
content:center;display:flex;flex-wrap:wrap;margin-
left: auto;margin-right: auto;">
  <div class="column">
    <div class="card" style="min-height:180px">
      <h3>CLIENT 1</h3>
      <p>RoomID:1234</p>
      <a class="card"  style="border-radius: 10px;text-
decoration: none;color: grey;" href="/chat/client1/1234"><i class="f
a fa-comments" style="font-size:36px"></i></a>
    </div>
  </div>
  <div class="column">
    <div class="card" style="min-height:180px">
      <h3>CLIENT 2</h3>
      <p>RoomID:5678</p>
      <a class="card"  style="border-radius: 10px;text-
decoration: none;color: grey;" href="/chat/client2/5678"><i class="f
a fa-comments" style="font-size:36px"></i></a>
    </div>
  </div>

  <div class="column">
    <div class="card" style="min-height:180px">
      <h3>CLIENT 3</h3>
      <p>RoomID:6789</p>
      <a class="card"  style="border-radius: 10px;text-
decoration: none;color: grey;" href="/chat/client3/6789"><i class="f
a fa-comments" style="font-size:36px"></i></a>
    </div>
  </div>


</div>

</body>
</html>
```

serverchat.html

```
<html>
    <head> <link rel="stylesheet" href="https://stackpath.bootstrapc
dn.com/bootstrap/4.5.2/css/bootstrap.min.css">
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.
4.1/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3
Og8ifwB6" crossorigin="anonymous"></script>
        <link href="https://fonts.googleapis.com/css2?family=Montserr
at:wght@300&display=swap" rel="stylesheet">
```

```
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/li
bs/font-awesome/4.7.0/css/font-awesome.min.css">
      <meta name="viewport" content="width=device-width, initial-
scale=1">

        <script type="text/javascript" src="//code.jquery.com/jquery-
1.4.2.min.js"></script>
        <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax
/libs/socket.io/1.3.6/socket.io.min.js"></script>
        <script type="text/javascript" charset="utf-8">
            var socket;
            $(document).ready(function(){
                socket = io.connect('http://' + document.domain + ':' + lo
cation.port + '/chat');
                socket.on('connect', function() {
                    socket.emit('joined', {});
                });
                socket.on('status', function(data) {
                    $('#chat').val($('#chat').val() + '<' + data.msg + '>\n');
                    $('#chat').scrollTop($('#chat')[0].scrollHeight);
                });
                socket.on('message', function(data) {
                    $('#chat').val($('#chat').val() + data.msg + '\n');
                    $('#chat').scrollTop($('#chat')[0].scrollHeight);
                });
                $('#text').keypress(function(e) {
                    var code = e.keyCode || e.which;
                    if (code == 13) {
                        text ='SERVER: '+ $('#text').val();
                        $('#text').val('');
                        socket.emit('text', {msg: text});
                    }
                });
            });
            function leave_room() {
                socket.emit('left', {}, function() {
                    socket.disconnect();
                    window.location.href = "{{ url_for('server') }}";
                });
            }
        </script>
    </head>
    <style>
        .card {
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
  padding: 16px;
  text-align: center;
  background-color: #f1f1f1;
}
</style>
    </style>
    <body style="padding: 20px;font-family: 'Montserrat', sans-
serif;font-size:14px;">
        <div class="card" style="padding: 20px;border-radius: 20px;">
        <h4>CHAT ROOM</h1>
        <textarea id="chat" cols="80" rows="20"></textarea><br><br>
        <input id="text" size="80" placeholder="Enter Here..."><br><br>
        <a  class="card"  style="text-
decoration: none;color: black;border-
radius: 20px;;"href="#" onclick="leave_room();">Terminate</a>
        </div>
    </body>
</html>
```

server.html

```
<!DOCTYPE html>
<html>
<head>
    <link href="https://fonts.googleapis.com/css2?family=Montserrat
:wght@300&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9M
uhOf23Q9Ifjh" crossorigin="anonymous">
    <script src="https://kit.fontawesome.com/86a929fcb7.js" crossori
gin="anonymous"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs
/font-awesome/4.7.0/css/font-awesome.min.css">
```

```
<meta name="viewport" content="width=device-width, initial-
scale=1">
<style>
* {
  box-sizing: border-box;
}

body {font-family: 'Montserrat', sans-serif;font-size:14px;
}

.column {
  float: left;
  width: 25%;
  max-width:300px;
  padding: 0 10px;
}

.row {margin: 0 -5px;}

.row:after {
  content: "";
  display: table;
  clear: both;
}
@media screen and (max-width: 600px) {
  .column {
    width: 100%;
    display: block;
    margin-bottom: 20px;
  }
}

.card {
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
  padding: 16px;
  text-align: center;
  background-color: #f1f1f1;
}
</style>
</head>
<nav class="navbar navbar-light bg-light">
```

```
    <a class="navbar-brand" href="#">
        <div class="display-
7">TECH SUPPORT - HELPDESK SOLUTION PANEL</div>
    </a>
  </nav>
<body style="text-align:center;background-color:azure">
    <i class="fa fa-comments-o" style="font-size:300px"></i>
    <h3 style="text-align: center;padding-
top: 30px;;">ACCEPT/IGNORE CLIENT REQUESTS</h3>
    <div class="row" style="margin-top:30px;justify-
content:center;display:flex;flex-wrap:wrap;margin-
left: auto;margin-right: auto;">
  {% for server in serv %}
    <div class="flex-container" style="display:flex;flex-wrap: wrap;">
    <div class="card" style="margin:10px;height:300px;">
      <h3>SERVER REQUEST</h3>
      <p>Candidate: {{server.name}}</p>
      {% if server.availability !=0 %}
      <a class="card"  style="border-radius: 10px;text-
decoration: none;color: grey;" href="/serverchat/{{server.id}}/{{serv
er.availability}}"><i class="fa fa-comments" style="font-
size:36px"></i></a>

      <a class="card"  style="margin-top:2px;border-radius: 10px;text-
decoration: none;color: grey;" href="/ignore/{{server.id}}">IGNORE
</a>

      {% endif %}
      {% if server.availability ==0 %}
      <a class="card"  style="border-radius: 10px;text-
decoration: none;color: grey;" >No Client Assigned</a>
      {% endif %}
      </div>
  </div>
{% endfor %}
</div>
</body>
</html>
```