

SUMMER TRAINING REPORT
ON
[RANDOM CHAT APPLICATION]
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE
AWARD
OF THE DEGREE OF
BACHELOR OF ENGINEERING
(Computer Science & Engineering)



APRIL-JULY,2021

SUBMITTED BY:

NAME : PRAJWAL SHARMA

UNIVERSITY UID : 19BCS1144

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CHANDIGARH UNIVERSITY GHARUAN, MOHALI

**CERTIFICATE BY COURSE WEBSITE / COMPANY /
INDUSTRY / INSTITUTE**



Codes Lipi Infotech Pvt. Ltd

Development | Training | Placement

(Regd. by MCA, Govt. Of India)

CIN: U80904HR2017PTC070031

Ref: CLI/JULY/2021/1557

Date: 10 July, 2021

TO WHOMSOEVER IT MAY CONCERN

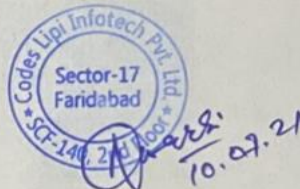
This is to certify that **Prajwal Sharma** has done his Training in **Python** from **Codes Lipi InfoTech Pvt. Ltd., Faridabad** from 10 April 2021 to 10 July 2021.

The project work entitled as "**Chat Application using Flask**", **Prajwal** has worked under the guidance of **Mr. Nitin Bisht**.

During the Training, Organization found him a good performer and excellent learner.
We wish him all the best in his future endeavors.

For: **Codes Lipi InfoTech Pvt. Ltd.**

Authorized Signatory



Regd. Office:-SCF-140,Second Floor,Sector-17,Faridabad-121002 (Hr.)
Ph: 0129-4000125 Email:info@codeslipiinfotech.com

Website: www.codeslipiinfotech.com

Instructions

Name : Python

Issuing Organization : Codes Lipi InfoTech Pvt. Ltd.

Issue Date : 10th July, 2021

Expiration Date : *This certification does not expire*

CANDIDATE'S DECLARATION

I PRAJWAL SHARMA hereby declare that I have undertaken Summer Training in Python during a period from 10th April, 2021 to 10th July, 2021 in partial fulfillment of requirements for the award of degree of B.E (COMPUTER SCIENCE & ENGINEERING) at CHANDIGARH UNIVERSITY GHARUAN, MOHALI. The work which is being presented in the training report submitted to Department of Computer Science & Engineering at CHANDIGARH UNIVERSITY GHARUAN, MOHALI is an authentic record of training work.

Signature of the Student

The training Viva–Voce Examination of Institutional training has been held on 01/12/2021 and accepted.

Signature of Internal Examiner

Signature of External Examiner

Contents

1. INTRODUCTION

- a. About Python, Flask, Database (PostgreSQL), HTML, CSS, Web Sockets
- b. Feasibility Study
- c. Methodology
- d. Hardware & Software Requirements
- e. About Project

2. IMPLEMENTATION

3. FINAL CODES

- a. Code of Html & JS Part
- b. Code of CSS Part
- c. Code of Flask (Python) Part
- d. Code of Database Part

ACKNOWLEDGEMENT

In the present world of competition there is a race of existence in which those are having will to come forward succeed. Project is like a bridge between theoretical and practical working. With this willing I joined this particular project. I would like to express my special thanks of gratitude to my college Chandigarh University who gave me the golden opportunity to do this wonderful project of **Random Chat Application**. I came to know about so many new things I am really thankful to them.

PRAJWAL SHARMA

19BCS1144

CSE 1 C

INTRODUCTION

Python :

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

This project is coded in python 3.9.6.

Flask :

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks. Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

Database (PostgreSQL) :

In computing, a database is an organized collection of data stored and accessed electronically from a computer system. Where databases are more complex, they are often developed using formal design and modelling techniques.

The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used loosely to refer to any of the DBMS, the database system or an application associated with the database.

Computer scientists may classify database-management systems according to the database models that they support. Relational databases became dominant in the 1980s. These model data as rows and columns in a series of tables, and the vast majority use SQL for writing and querying data. In the 2000s, non-relational databases became popular, referred to as NoSQL because they use different query languages.

In this project, a type of SQL is used i.e., **PostgreSQL**.

PostgreSQL also known as Postgres, is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley. In 1996, the project was renamed to PostgreSQL to reflect its support for SQL. After a review in 2007, the development team decided to keep the name PostgreSQL and the alias Postgres.

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. It is the default database for macOS Server and is also available for Windows, Linux, FreeBSD, and OpenBSD.

In this project I used some basic queries like, **CREATE, INSERT, SELECT**.

HTML :

The **Hyper Text Markup Language**, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

CSS :

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

Web Sockets :

The WebSocket specification defines an API establishing “socket” connections between a web browser and a server. In plain words: there is a persistent connection between the client and the server and both parties can start sending data at any time. The client establishes a WebSocket connection through a process known as the WebSocket handshake. This

process starts with the client sending a regular HTTP request to the server. An Upgrade header is included in this request which informs the server that the client wishes to establish a WebSocket connection. Now that the handshake is complete the initial HTTP connection is replaced by a WebSocket connection that uses the same underlying TCP/IP connection. At this point, either party can start sending data.

With Web Sockets, you can transfer as much data as you like without incurring the overhead associated with traditional HTTP requests. Data is transferred through a WebSocket as messages, each of which consists of one or more frames containing the data you are sending (the payload). In order to ensure the message can be properly reconstructed when it reaches the client each frame is prefixed with 4–12 bytes of data about the payload. Using this frame-based messaging system helps to reduce the amount of non-payload data that is transferred, leading to significant reductions in latency.

Feasibility Study :

Feasibility study can help you determine whether or not you should proceed with your project. It is essential to evaluate cost and benefit. It is essential to evaluate cost and benefit of the proposed system.

Four types of feasibility study are taken into consideration.

1. Technical feasibility: It includes finding out technologies for the project, both hardware and software. For Chat Application, user must have internet connection. While using Application, make sure you have a steady internet connection. It is also not an issue in this era where almost every home or office has Wi-Fi.

2. Operational feasibility: It is the ease and simplicity of operation of proposed system. System does not require any special skill set for users to operate it. In fact, it is designed to be used by almost everyone. Kids who still don't know to write can read out problems for system and get answers.

3. Economic feasibility: Here, we find the total cost and benefit of the proposed system over current system. For this project, the main cost is documentation cost. User also would have to pay for internet connection. As far as maintenance is concerned, it won't cost too much.

4. Organizational feasibility: This shows the management and organizational structure of the project. This project is not built by a team. The management tasks are all to be carried out by a single person. That won't create any management issues and will increase the feasibility of the project.

This project is technically feasible with no external hardware requirements. Also, it is simple in operation and does not cost training or repairs. Overall feasibility study of the project reveals that the goals of the proposed system are achievable. Decision is taken to proceed with the project.

Methodology :

This project contains three steps:

1. Development of Frontend part using **HTML, CSS**.
2. Coding and Development of Backend part using **FLASK (Python)** and **Web Sockets** using both FLASK and JS.
3. Making and Establishing the **Database** connection.

Hardware & Software Requirements :

The software is designed to be light-weighted so that it doesn't be a burden on the machine running it. This system is being build keeping in mind the generally available hardware and software compatibility. Here are the minimum hardware and software requirement for speaking assistant.

Hardware:

- Pentium-pro processor or later.
- RAM 512MB or more.

Software:

- Windows 7(32-bit) or above.
- Python 3.0 or later

About Project :

Random Chat Application is a type of web application in which user can talk to random people with anonymous name.

This application contains **Sign Up Page, Login Page, Room Joining/Creating Page** and **Messaging Page**.

SIGN UP PAGE

Sign up Form

Enter your email :

Create Username :

Create Password :

Submit

Already a user? Login here :

Login

LOGIN PAGE

Login Form

Username :

Password :

Submit

Not a user? Sign up here :

Sign up

ROOM JOINING/CREATING PAGE

Logout

Welcome to Uniquedu chat app

Enter username :

Enter room :

Enter

MESSAGING PAGE

Leave Chat

Welcome to chat room python

Be the first to start chat...

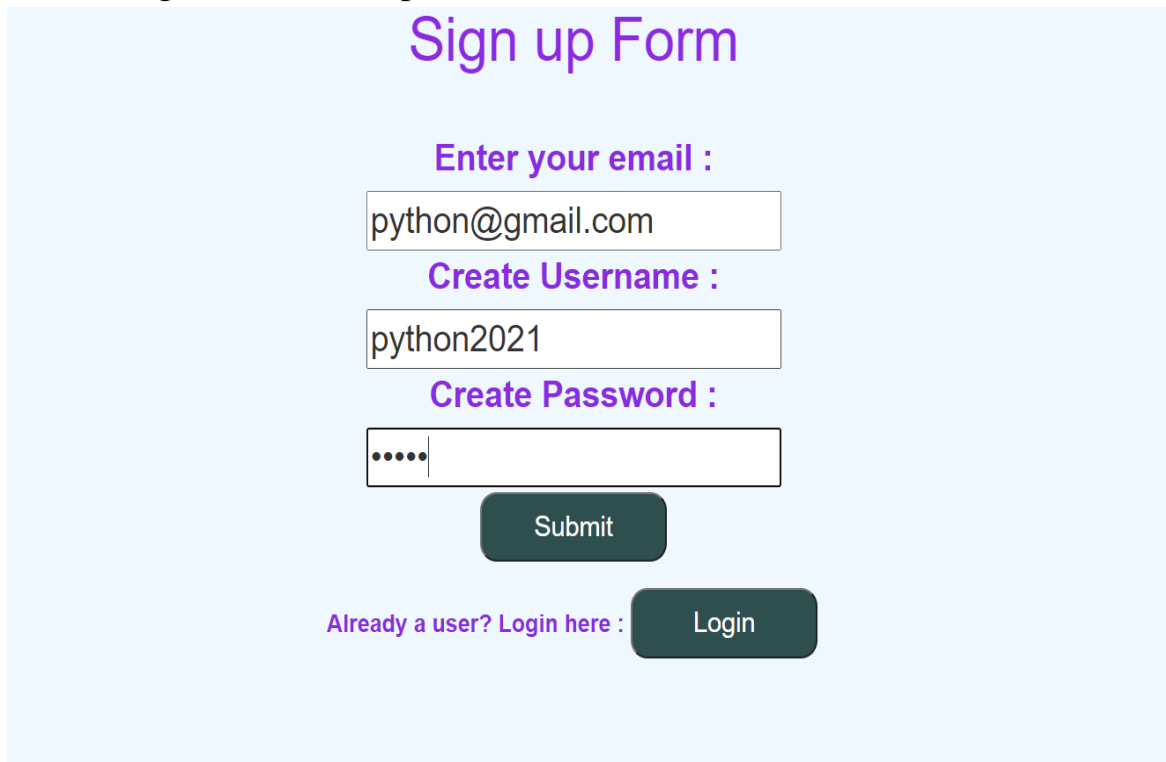
Enter your message here

Send

IMPLEMENTATION

As mentioned in **ABOUT PROJECT** section, this application connects random people with each other and they can talk without actually showing their names.

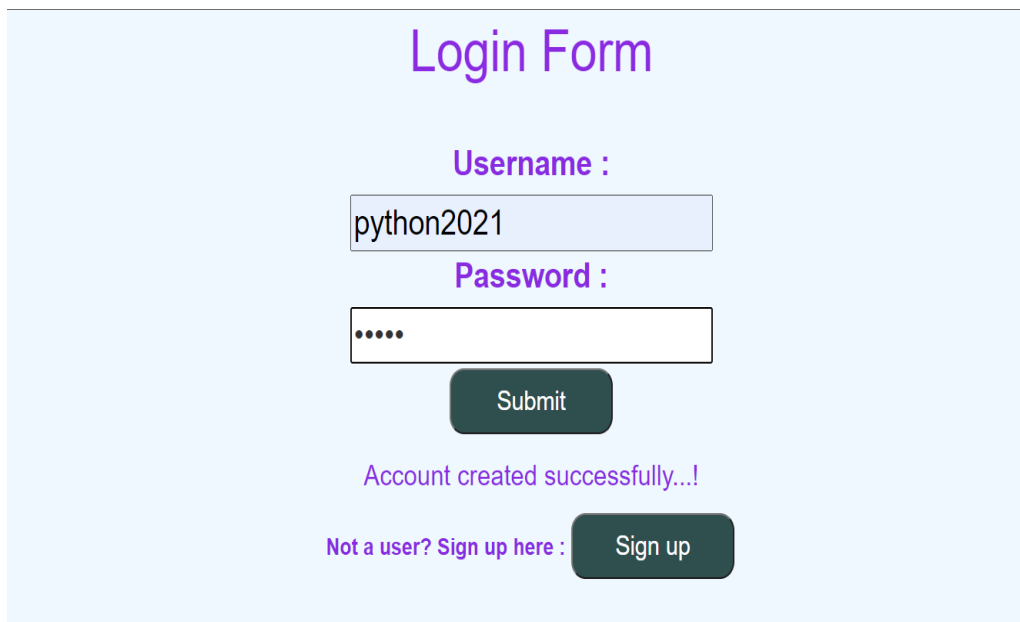
1. Firstly, we have to register ourself by simply entering email id and creating username and password.



The image shows a 'Sign up Form' with a light blue background. At the top, the title 'Sign up Form' is written in purple. Below it, the label 'Enter your email :' is in purple, followed by a text input field containing 'python@gmail.com'. The next label is 'Create Username :' in purple, followed by a text input field containing 'python2021'. The third label is 'Create Password :' in purple, followed by a password input field with four dots. Below these fields is a dark green 'Submit' button. At the bottom, there is a link 'Already a user? Login here :' in purple, followed by a dark green 'Login' button.

Here you can see that if we are already a user, we can simply go to login page from here also.

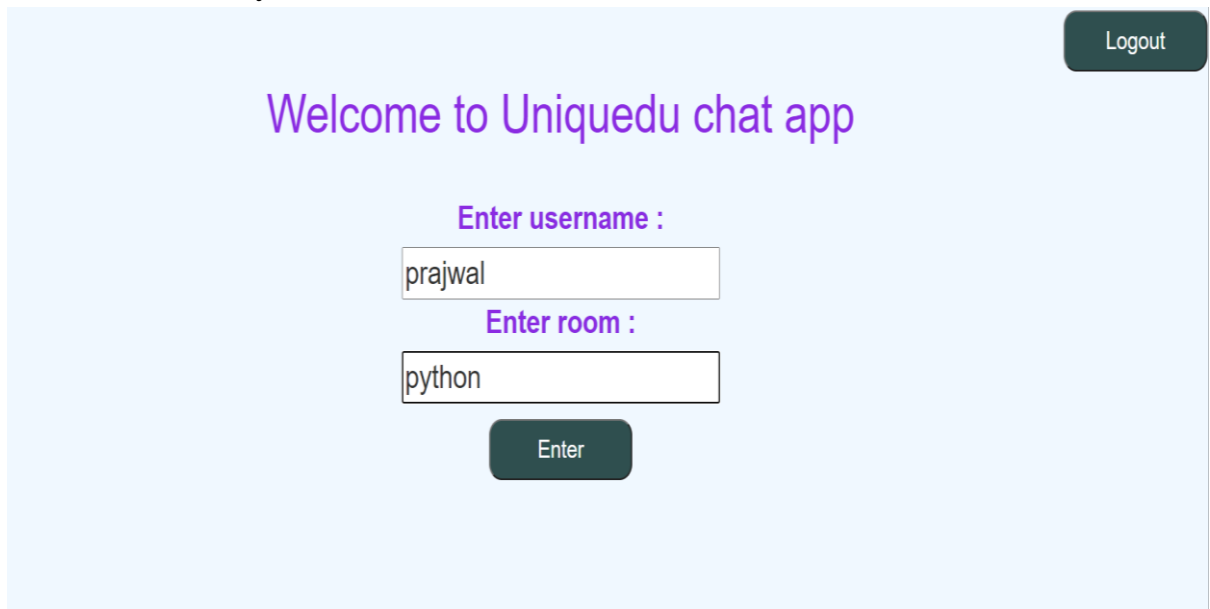
2. Once we have registered, now we have to login to the registered account.



The image shows a 'Login Form' with a light blue background. At the top, the title 'Login Form' is written in purple. Below it, the label 'Username :' is in purple, followed by a text input field containing 'python2021'. The next label is 'Password :' in purple, followed by a password input field with four dots. Below these fields is a dark green 'Submit' button. At the bottom, there is a message 'Account created successfully...!' in purple. Below that, there is a link 'Not a user? Sign up here :' in purple, followed by a dark green 'Sign up' button.

Here you can see the message of account creation, which comes just after registration. And also, we can redirect to signup page again directly from here.

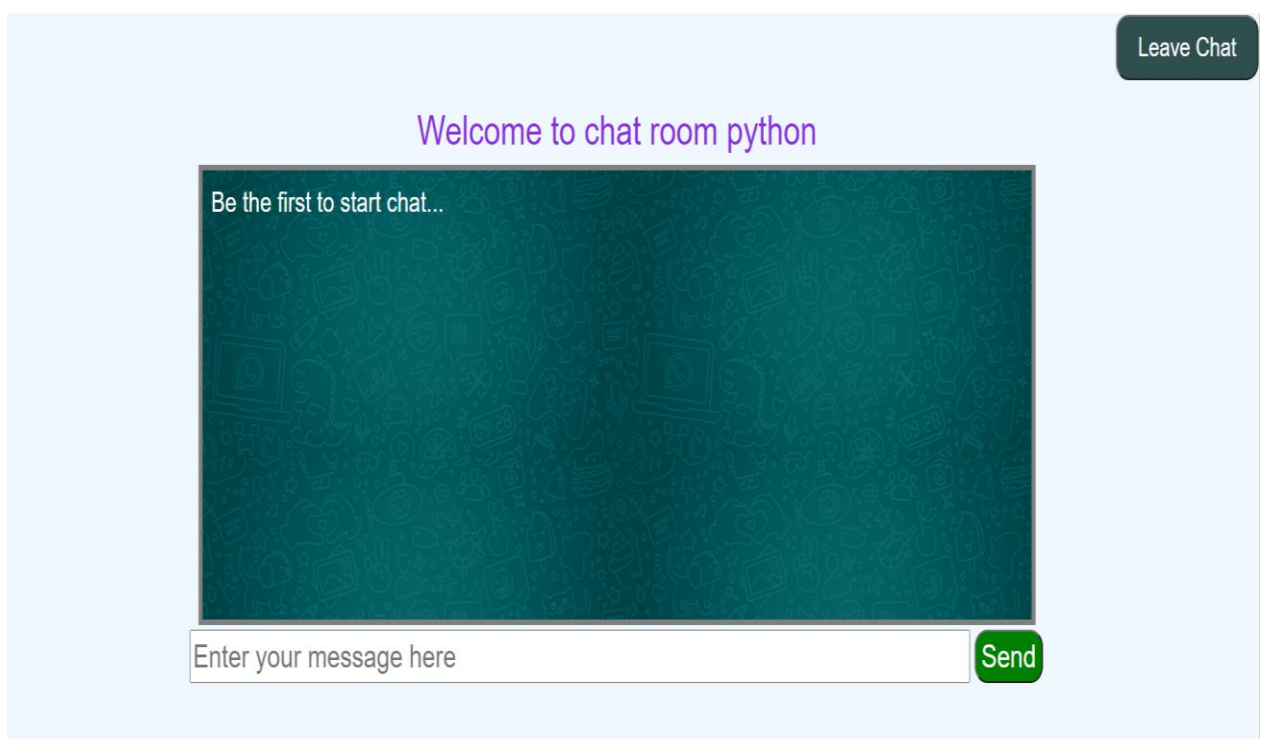
3. Now we have login successfully and now we can create or join any room with any name as our wish.



The screenshot shows a light blue background with a dark blue 'Logout' button in the top right corner. The main heading is 'Welcome to Uniquedu chat app' in purple. Below it, there are two input fields: 'Enter username :' with the text 'prajwal' and 'Enter room :' with the text 'python'. A dark blue 'Enter' button is positioned below the room input field.

As you can see that we can easily logout from here by clicking the logout button.

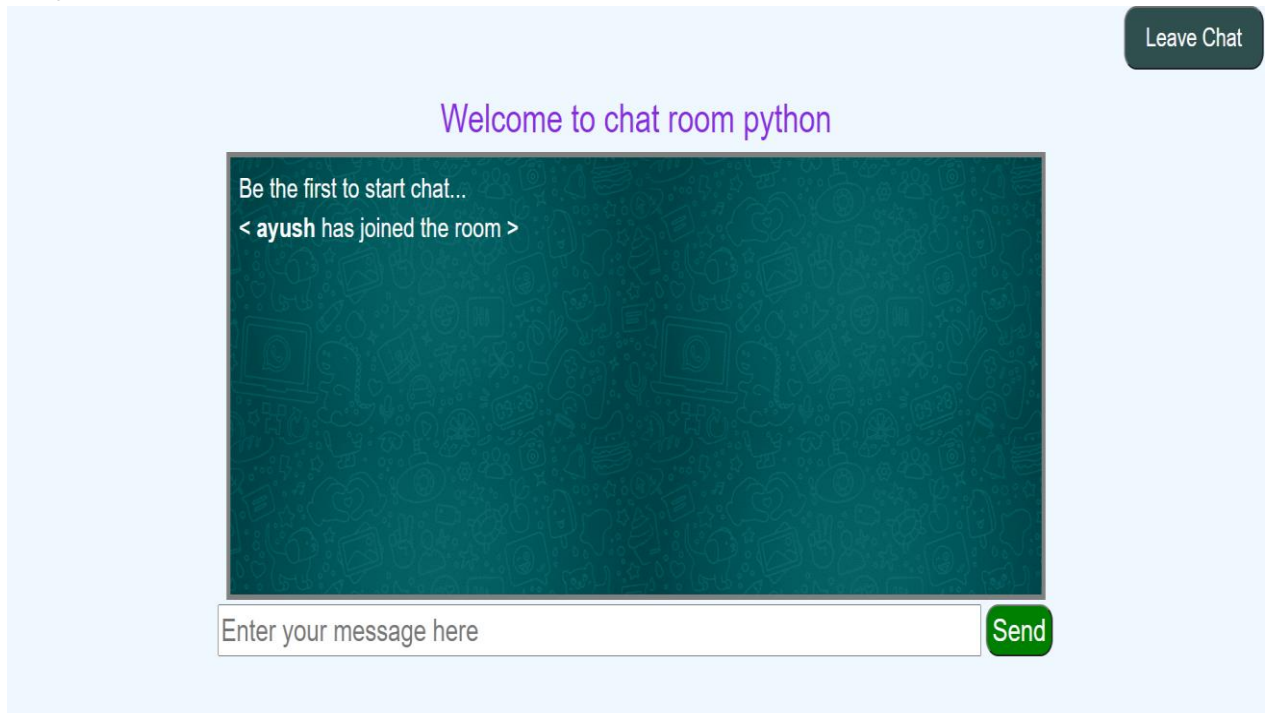
4. After entering the room, we can chat with random people as many as we can.



The screenshot shows a light blue background with a dark blue 'Leave Chat' button in the top right corner. The main heading is 'Welcome to chat room python' in purple. Below it is a large dark green rectangular area with a pattern of small icons and the text 'Be the first to start chat...'. At the bottom, there is a text input field with the placeholder 'Enter your message here' and a green 'Send' button.

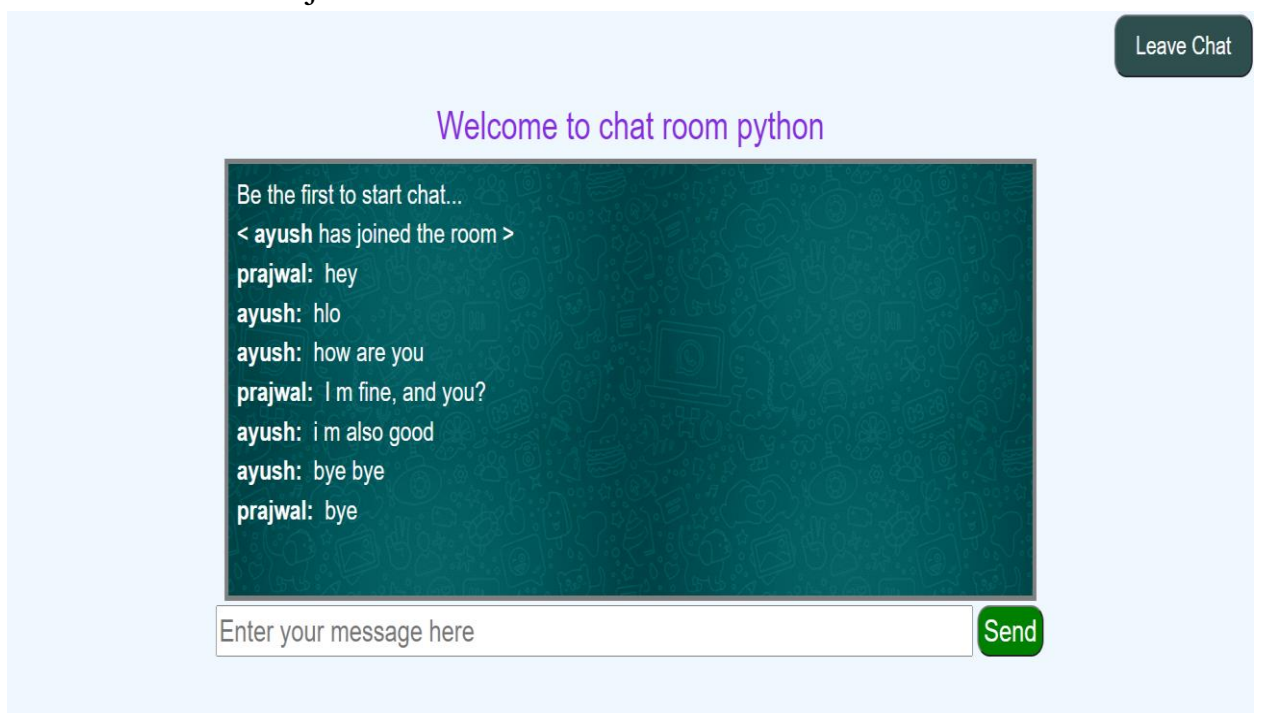
Now I will create a new account and join the same and then we can see that both can chat with each other.

I have joined the same room named python with username Ayush and we can see that there is a message of joining of Ayush in the chat window of Prajwal.



Now they both can talk to each other.

Chat window of Prajwal :



Chat window of Ayush :

[Leave Chat](#)

Welcome to chat room python

Be the first to start chat...

prajwal: hey

ayush: hlo

ayush: how are you

prajwal: I m fine, and you?

ayush: i m also good

ayush: bye bye

prajwal: bye

< prajwal has left the room >

[Send](#)

Prajwal leaves the room first so, there is a message of room leaving in the chat window of Ayush.

This was the implementation of my project. Now, I want to show some more things that my application does.

1. Checking of user existence.

Sign up Form

Enter your email :

Create Username :

Create Password :

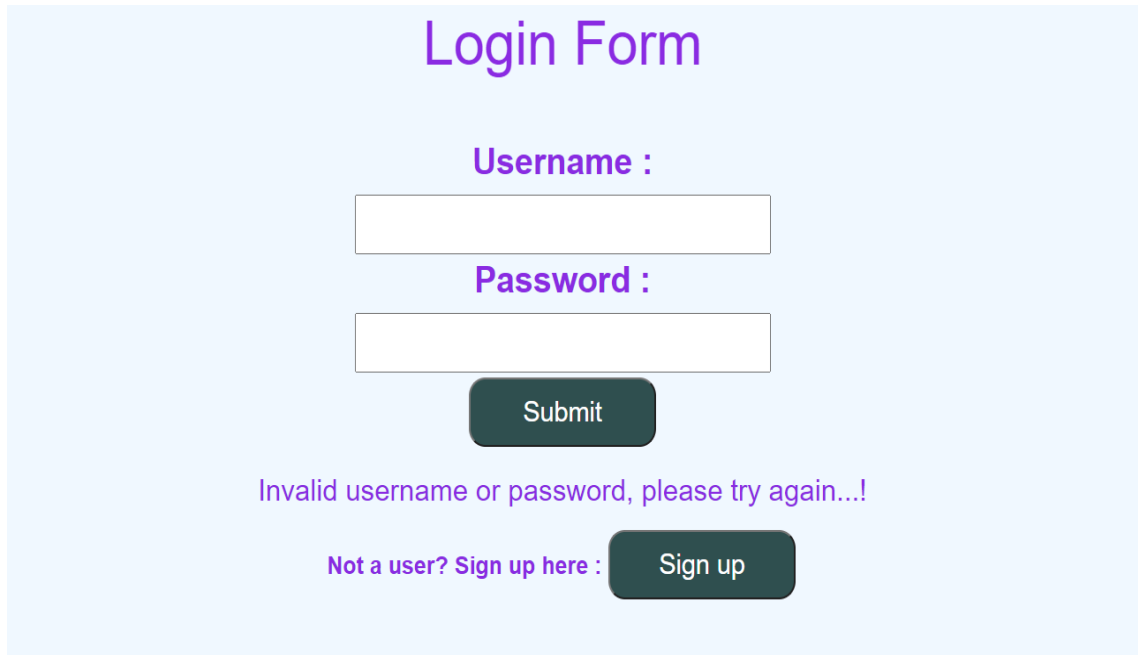
[Submit](#)

Username already exists...!

Already a user? Login here : [Login](#)

If user already exists than this application does not allows to again create the account with same name.

2. Checking of username and password during login.



Login Form

Username :

Password :

Submit

Invalid username or password, please try again...!

Not a user? Sign up here : Sign up

If username is not present or password is wrong than above displayed message is shown.

FINAL CODES

Code of HTML & JS Part :

Signup Page :

```
1. <html>
2.   <head>
3.     <meta charset="UTF-8">
4.     <link rel="stylesheet" href="../static/styles.css">
5.     <link rel="stylesheet"
6.       href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
7.     <script
8.       src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
9.     <script
10.      src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
11.    <title>Sign up</title>
12.  </head>
13.  <body>
14.    <div class="visible-lg" style="font-size: 50px; text-align: center; color:
15.      blueviolet;">Sign up Form</div>
16.    <div class="hidden-lg" style="font-size: 70px; text-align: center; color:
17.      blueviolet">
18.      <br><br><br><br><br>
19.    </div>
20.  </body>
21. </html>
```

```
14.         Sign up Form
15.     </div>
16.     <br><br>
17.     <div class="visible-lg" style="text-align: center;">
18.         <form action="/signup" method="POST">
19.             <label style="color: BlueViolet; font-size: 30px;">Enter your email
20.             : </label><br>
21.             <input type="text" name="email" style="font-size: 30px;"
22.             required><br>
23.             <label style="color: BlueViolet; font-size: 30px;">Create Username
24.             : </label><br>
25.             <input type="text" name="username" style="font-size: 30px;"
26.             required><br>
27.             <label style="color: BlueViolet; font-size: 30px;">Create Password
28.             : </label><br>
29.             <input type="password" name="password" style="font-size: 30px;"
30.             required><br>
31.             <input type="submit" value="Submit" class="b1">
32.         </form>
33.     </div>
34.     <div class="hidden-lg" style="text-align: center;">
35.         <form action="/signup" method="POST">
36.             <label style="color: BlueViolet; font-size: 60px;">Enter your email
37.             : </label><br>
38.             <input size="25" type="text" name="email" style="font-size: 40px;"
39.             required><br>
40.             <label style="color: BlueViolet; font-size: 60px;">Create Username
41.             : </label><br>
42.             <input size="25" type="text" name="username" style="font-size:
43.             40px;" required><br>
44.             <label style="color: BlueViolet; font-size: 60px;">Create Password
45.             : </label><br>
46.             <input size="25" type="password" name="password" style="font-size:
47.             40px;" required><br>
48.             <input type="submit" value="Submit" class="b1">
49.         </form>
50.     </div>
```

```

45.
46.     </form>
47.     <div style="text-align: center; font-size: 25px; color: blueviolet;"
      class="visible-lg">
48.         <p>{{msg}}</p>
49.     </div>
50.     <div style="text-align: center; font-size: 30px; color: blueviolet;"
      class="hidden-lg">
51.         <p>{{msg}}</p>
52.     </div>
53.     <form action="/tologin" style="text-align: center;">
54.         <div class="visible-lg">
55.             <label style="color: BlueViolet; font-size: 20px;">Already a user?
      Login here :</label>
56.             <input type="submit" value="Login" class="b1">
57.         </div>
58.         <div class="hidden-lg">
59.             <label style="color: BlueViolet; font-size: 30px;">Already a user?
      Login here :</label>
60.             <input type="submit" value="Login" class="b1">
61.         </div>
62.     </form>
63. </body>
64. </html>

```

Login Page :

```

2. <html>
3.     <head>
4.         <meta charset="UTF-8">
5.         <link rel="stylesheet" href="../static/styles.css">
6.         <link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
7.         <script
      src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
8.         <script
      src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
9.         <title>Login</title>
10.    </head>
11.    <body>
12.        <div class="visible-lg" style="font-size: 50px; text-align: center; color:
      blueviolet;">Login Form</div>
13.        <div class="hidden-lg" style="font-size: 70px; text-align: center; color:
      blueviolet">

```

```
14.         <br><br><br><br><br>
15.         Login Form
16.     </div>
17.     <br><br>
18.     <div class="visible-lg" style="text-align: center;">
19.         <form action="/login" method="POST">
20.             <label style="color: BlueViolet; font-size: 30px;">Username :
21.         </label><br>
22.             <input type="text" name="username" style="font-size: 30px;"
23.             required><br>
24.             <label style="color: BlueViolet; font-size: 30px;">Password :
25.         </label><br>
26.             <input type="password" name="password" style="font-size: 30px;"
27.             required><br>
28.             <input type="submit" value="Submit" class="b1">
29.         </form>
30.     </div>
31.     <div class="hidden-lg" style="text-align: center;">
32.         <form action="/login" method="POST">
33.             <label style="color: BlueViolet; font-size: 60px;">Username :
34.         </label><br>
35.             <input size="25" type="text" name="username" style="font-size:
36.             40px;" required><br>
37.             <label style="color: BlueViolet; font-size: 60px;">Password :
38.         </label><br>
39.             <input size="25" type="password" name="password" style="font-size:
40.             40px;" required><br>
41.             <input type="submit" value="Submit" class="b1">
42.         </form>
43.     </div>
44.     <div style="text-align: center; font-size: 25px; color: blueviolet;"
45.     class="visible-lg">
46.         <p>{{msg}}</p>
47.     </div>
48.     <div style="text-align: center; font-size: 30px; color: blueviolet;"
49.     class="hidden-lg">
50.         <p>{{msg}}</p>
51.     </div>
52.     <form action="/" style="text-align: center;">
```

```

47.         <div class="visible-lg">
48.             <label style="color: BlueViolet; font-size: 20px;">Not a user? Sign
up here :</label>
49.             <input type="submit" value="Sign up" class="b1">
50.         </div>
51.         <div class="hidden-lg">
52.             <label style="color: BlueViolet; font-size: 30px;">Not a user? Sign
up here :</label>
53.             <input type="submit" value="Sign up" class="b1">
54.         </div>
55.     </form>
56. </body>
57.</html>

```

Room Joining/Creating Page :

```

3. <!DOCTYPE html>
4. <html lang="en">
5.     <head>
6.         <meta charset="UTF-8">
7.         <title>Create/Join chat room</title>
8.         <link rel="stylesheet" href="../static/styles.css">
9.         <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
10.        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
11.        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
12.    </head>
13.    <body>
14.        <div class="visible-lg">
15.            <form action="/logout" style="text-align: right;">
16.                <input type="submit" value="Logout" class="b1">
17.            </form>
18.        </div>
19.        <div class="visible-lg" style="font-size: 50px; text-align: center; color:
blueviolet;">Welcome to Uniquedu chat app</div>
20.        <div class="hidden-lg" style="font-size: 60px; text-align: center; color:
blueviolet">
21.            <br><br><br><br><br>
22.            Welcome to Uniquedu chat app
23.        </div>
24.        <br><br>

```

```
25.         <div class="visible-lg" style="text-align: center;">
26.             <form action="/chat">
27.                 <label style="color: BlueViolet; font-size: 30px;">Enter username
28.                 :</label><br>
29.                 <input type="text" name="username" style="font-size: 30px;"
30.                 required><br>
31.                 <label style="color: BlueViolet; font-size: 30px;">Enter room
32.                 :</label><br>
33.                 <input type="text" name="room" style="font-size: 30px;" required>
34.                 <div style="text-align: center; font-size: 25px; color:
35.                 blueviolet;">
36.                     <p>{{msg}}</p>
37.                 </div>
38.                 <input type="submit" value="Enter" class="b1">
39.             </form>
40.         </div>
41.         <div class="hidden-lg" style="text-align: center;">
42.             <form action="/chat">
43.                 <label style="color: BlueViolet; font-size: 60px;">Enter username
44.                 :</label><br>
45.                 <input size="25" type="text" name="username" style="font-size:
46.                 40px;" required><br>
47.                 <label style="color: BlueViolet; font-size: 60px;">Enter room
48.                 :</label><br>
49.                 <input size="25" type="text" name="room" style="font-size: 40px;"
50.                 required>
51.                 <div style="text-align: center; font-size: 30px; color:
52.                 blueviolet;">
53.                     <p>{{msg}}</p>
54.                 </div>
55.                 <input type="submit" value="Enter" class="b1">
56.             </form>
57.         </div>
58.         <br>
59.         <div class="hidden-lg">
60.             <form action="/logout" style="text-align: center;">
61.                 <input type="submit" value="Logout" class="b1">
62.             </form>
63.         </div>
```

```
59.     </body>
60.</html>
```

Messaging Window :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>{{room}} room chats</title>
    <link rel="stylesheet" href="../static/styles.css">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
  </head>
  <body>
    <form action="/leave" style="text-align: right;">
      <input type="submit" value="Leave Chat" class="b1">
    </form>
    <h1>Welcome to chat room {{ room }}</h1>

    <div id="messages" class="msg_visible visible-lg">
      {{no_msg}}
      {% for message in messages %}
        <b>{{message[0]}}</b> : {{message[1]}}<br>
      {% endfor %}
    </div>

    <div id="message" class="msg_hidden hidden-lg">
      {{no_msg}}
      {% for message in messages %}
        <b>{{message[0]}}</b> : {{message[1]}}<br>
      {% endfor %}
    </div>

    <form id="message_input_form" style="text-align: center;">
      <input type="text" id="message_input" placeholder="Enter your message here" style="font-
size: 28px;" size="60" required>
      <input type="submit" value="Send" class="send">
    </form>
    <form id="message_input_forms" style="text-align: center;" class="hidden-lg">
      <input type="text" id="message_inputs" placeholder="Enter your message here" style="font-
size: 28px;" size="40" required>
      <input type="submit" value="Send" class="send">
    </form>
  </body>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js" integrity="sha512-
bLT0Qm9VnAYZDflyKcBaQ2gg0hSYNQrJ8RilYldYQ1FxBYoCLTUjuuRuZo+fqjhX/Qtq/1itJ0C2ejDxltZVFg=="
crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/3.0.4/socket.io.js"
integrity="sha512-
```

```
aMGMvNYu8Ue4G+fHa359jcPb1u+ytAF+P2SCb+PxrjCd03n3ZTxJ30zuH39rimUggmTwmh2u7wvQsDTHESnmfQ=="
crossorigin="anonymous"></script>
<script>
    const socket = io.connect("http://localhost:5000");

    socket.on('connect', function () {
        socket.emit('join_room', {
            username: "{{ username }}",
            room: "{{ room }}"
        });

        let message_input = document.getElementById('message_input');

        document.getElementById('message_input_form').onsubmit = function (e) {
            e.preventDefault();
            let message = message_input.value.trim();
            if (message.length) {
                socket.emit('send_message', {
                    username: "{{ username }}",
                    room: "{{ room }}",
                    message: message
                })
            }
            message_input.value = '';
            message_input.focus();
        }

        let message_inputs = document.getElementById('message_inputs');

        document.getElementById('message_input_forms').onsubmit = function (e) {
            e.preventDefault();
            let messages = message_inputs.value.trim();
            if (messages.length) {
                socket.emit('send_message', {
                    username: "{{ username }}",
                    room: "{{ room }}",
                    messages: messages
                })
            }
            message_inputs.value = '';
            message_inputs.focus();
        }
    });

    window.onbeforeunload = function () {
        socket.emit('leave_room', {
            username: "{{ username }}",
            room: "{{ room }}"
        })
    };

    socket.on('receive_message', function (data) {
        console.log(data);
    });

```



```

    const newNode = document.createElement('div');
    newNode.innerHTML = `<b>${data.username}&nbsp;</b> ${data.message}`;
    document.getElementById('messages').appendChild(newNode);
  });

  socket.on('join_room_announcement', function (data) {
    console.log(data);
    if (data.username !== "{ username }") {
      const newNode = document.createElement('div');
      newNode.innerHTML = `< <b>${data.username}</b> has joined the room >`;
      document.getElementById('messages').appendChild(newNode);
    }
  });

  socket.on('leave_room_announcement', function (data) {
    console.log(data);
    const newNode = document.createElement('div');
    newNode.innerHTML = `< <b>${data.username}</b> has left the room >`;
    document.getElementById('messages').appendChild(newNode);
  });

  socket.on('receive_message', function (data) {
    console.log(data);
    const newNode = document.createElement('div');
    newNode.innerHTML = `<b>${data.username}&nbsp;</b> ${data.message}`;
    document.getElementById('message').appendChild(newNode);
  });

  socket.on('join_room_announcement', function (data) {
    console.log(data);
    if (data.username !== "{ username }") {
      const newNode = document.createElement('div');
      newNode.innerHTML = `< <b>${data.username}</b> has joined the room >`;
      document.getElementById('message').appendChild(newNode);
    }
  });

  socket.on('leave_room_announcement', function (data) {
    console.log(data);
    const newNode = document.createElement('div');
    newNode.innerHTML = `< <b>${data.username}</b> has left the room >`;
    document.getElementById('message').appendChild(newNode);
  });
</script>
</html>

```

Code of CSS Part :

```
body {
  background-color: #F0F8FF !important;
}

h1 {
  color: #8A2BE2 !important;
  text-align: center;
  font-size: 50px;
}

h2 {
  color: BlueViolet !important;
  text-align: center;
  font-size: 30px;
}

.b1 {
  background-color: darkslategray;
  color: white;
  text-align: center;
  display: inline-block;
  font-size: 23px;
  margin: 4px 2px;
  cursor: pointer;
  padding: 10px 20px;
  border-radius: 15px;
  transition-duration: 0.4s;
  width: 170px;
}

.b1:hover {
  color: darkslategray;
  background-color: white;
}

.send {
  background-color: green;
  color: white;
  text-align: center;
  display: inline-block;
  font-size: 28px;
  margin: 4px 2px;
  cursor: pointer;
  border-radius: 15px;
  transition-duration: 0.4s;
}

.send:hover {
  color: darkslategray;
  background-color: white;
}
```

```

}

.msg_visible {
    background-image: url("chat_bg.png");
    color: white;
    width: 1000px;
    height: 400px;
    padding: 10px;
    border: 5px solid gray;
    margin: auto;
    overflow: auto;
    font-size: 25px;
}

.msg_hidden {
    background-image: url("chat_bg.png");
    color: white;
    width: 900px;
    height: 1500px;
    padding: 10px;
    border: 5px solid gray;
    margin: auto;
    overflow: auto;
    font-size: 25px;
}

```

Code of FLASK(Python) Part :

app.py

```

from flask import Flask, render_template, request, redirect, url_for, request
from flask_socketio import SocketIO, join_room, leave_room
from flask_login import LoginManager, login_user, login_required, logout_user
import json
import db_connect as db

app = Flask(__name__)
app.secret_key = "prajwal sharma"
socketio = SocketIO(app, cors_allowed_origins="*")
login_manager = LoginManager()
login_manager.login_view = 'login'
login_manager.init_app(app)

@app.route('/')
def home():
    return render_template("signup.html")

#Redirect to login page
@app.route("/tologin")
def to_login() :
    return render_template("login.html")

```

```

@app.route('/login', methods = ['GET', 'POST'])
def login() :
    message = ''
    if request.method == 'POST' :
        username = request.form.get('username')
        password_input = request.form.get('password')
        file = 'username.json'
        with open(file, 'r') as r :
            username_list = json.load(r)
        if username not in username_list :
            message = "Invalid username or password, please try again...!"
        elif username in username_list :
            user = db.get_user(username)
            if user and user.check_password(password_input) :
                login_user(user)
                return render_template('index.html')
            else :
                message = "Invalid username or password, please try again...!"
    return render_template('login.html', msg = message)

```

```

@app.route('/signup', methods = ['GET', 'POST'])
def signup() :
    if request.method == 'POST' :
        username = request.form.get('username')
        email = request.form.get('email')
        password = request.form.get('password')
        file = 'username.json'
        with open(file, 'r') as r :
            username_list = json.load(r)

        if username not in username_list :
            username_list.append(username)
            with open(file, 'w') as w :
                json.dump(username_list, w, indent=4)
            db.save_user(username, password, email)
            return render_template('login.html', msg = "Account created successfully...!")
        elif username in username_list :
            return render_template('signup.html', msg = "Username already exists...!")

```

```

@app.route('/logout')
@login_required
def logout() :
    logout_user()
    return render_template('login.html', msg = "Logout successfully...!")

```

```

@app.route('/chat')
@login_required
def chat():
    username = request.args.get('username')
    room = request.args.get('room')

    if username and room:
        msg_data = db.get_message(room)

```

```

        if msg_data :
            return render_template('chat.html', username=username, room=room, messages = msg_data)
        else :
            return render_template('chat.html', username=username, room=room, no_msg = "Be the first
to start chat...")
    else:
        return redirect(url_for('to_login'))

@app.route('/leave')
def leave() :
    return render_template('index.html', msg = "Successfully exited from recent chat room...!")

@socketio.on('send_message')
def handle_send_message_event(data):
    app.logger.info("{} has sent message to the room {}".format(data['username'], data['room'], data['message']))
    db.save_message(data['room'], data['message'], data['username'])
    socketio.emit('receive_message', data, room=data['room'])

@socketio.on('join_room')
def handle_join_room_event(data):
    app.logger.info("{} has joined the room {}".format(data['username'], data['room']))
    join_room(data['room'])
    socketio.emit('join_room_announcement', data, room=data['room'])

@socketio.on('leave_room')
def handle_leave_room_event(data):
    app.logger.info("{} has left the room {}".format(data['username'], data['room']))
    leave_room(data['room'])
    socketio.emit('leave_room_announcement', data, room=data['room'])

@login_manager.user_loader
def load_user(username) :
    return db.get_user(username)

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

```

user.py

```

from werkzeug.security import check_password_hash

class User :
    def __init__(self, username, password, email) :
        self.username = username
        self.password = password
        self.email = email

    @staticmethod
    def is_authenticated() :
        return True

```

```

@staticmethod
def is_active() :
    return True

@staticmethod
def is_anonymous() :
    return False

def get_id(self) :
    return self.username

def check_password(self, password_input) :
    return check_password_hash(self.password, password_input)

```

Code of Database Part :

```

import psycopg2
from user import User
from werkzeug.security import generate_password_hash

connection = psycopg2.connect(database='postgres', user='postgres', password='admin',
host='localhost')
obj = connection.cursor()
print("Creating connection")

obj.execute(""" create table if not exists messages (
                room varchar(255),
                text varchar(255),
                sender varchar(255)
            ); """)
connection.commit()

def save_user(username, password, email) :
    hash_password = generate_password_hash(password)
    obj.execute(""" insert into chatapp values ('{0}', '{1}', '{2}') """.format(username,
hash_password, email))
    connection.commit()
    print("Save user Query executed")

def get_user(username) :
    obj.execute(""" select * from chatapp where username='{0}' """.format(username))
    data_user = obj.fetchall()
    user_data = data_user.pop()
    user_data = list(user_data)
    print("Get user Query executed")
    x = User(user_data[0], user_data[1], user_data[2]) if user_data else None
    return x

def save_message(room, text, sender) :
    obj.execute(""" insert into messages values ('{0}', '{1}', '{2}') """.format(room, text, sender))
    connection.commit()
    print("Save message Query executed")

```

```
def get_message(room) :  
    obj.execute(""" select sender, text from messages where room='{0}' """.format(room))  
    msg = obj.fetchall()  
    print("Get msg Query executed")  
    return msg
```