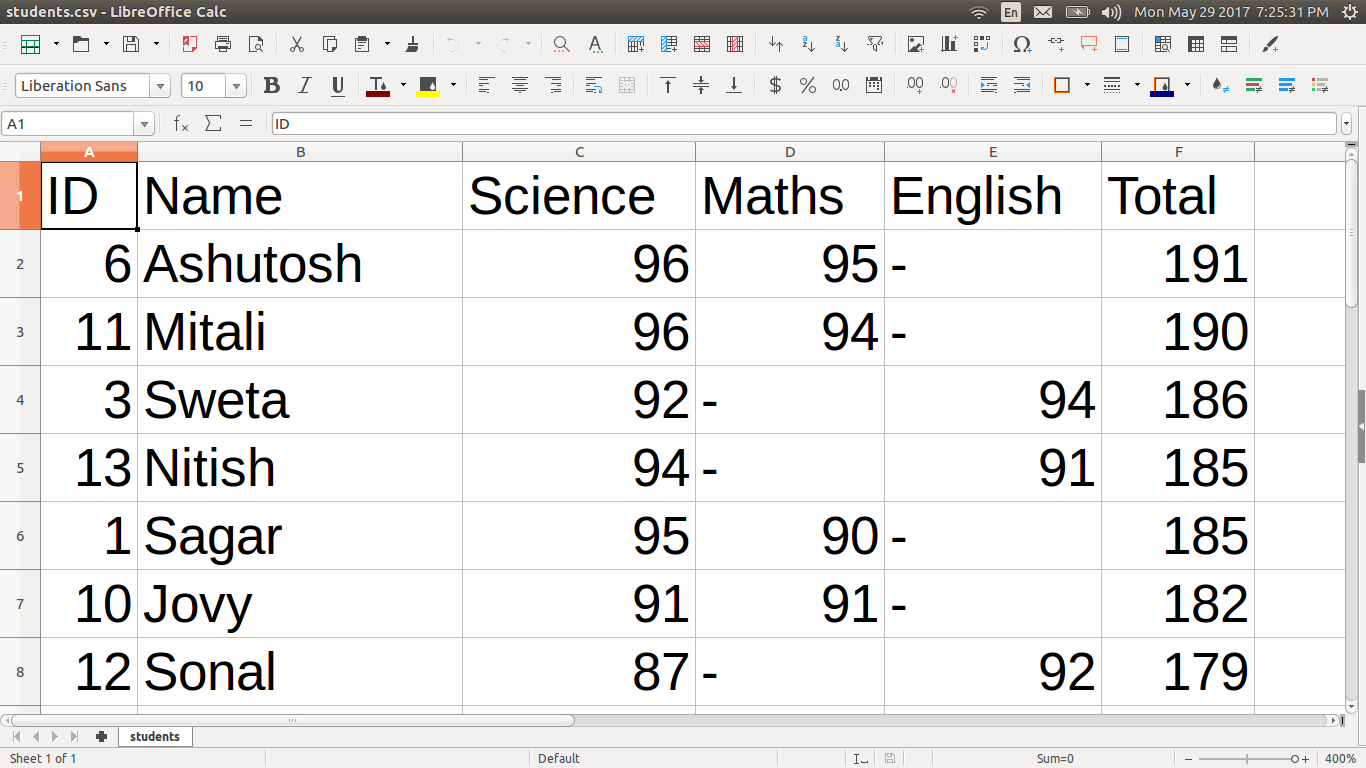
**Assignment 1:-**

1. Here the basic assumption is that the file is present in the same directory as the .py file for simplicity purposes.

2. We can either assume that the file named students.csv is completely empty containing only headers or it contains some data as shown above. Either way works.

3. The ID of student needs to be unique. If the student has enrolled for the subject then his marks are entered else **‘-’** is entered indicating that the student has not enrolled for the subject.

4. The data structure used to impelement LRU cache is OrderedDict which is a type of dictionary in Python but remembers the order in which the keys were inserted first which was perfect for LRU Cache implementation.

5. To run the file place the .py file and the students.csv file in the same directory. Open command line/terminal in the same directory. Run the command – **python lru\_cache.py**

**Assignment 2:-**

For running the application we need to install two dependencies first:

a. [\*sudo\*] pip install Flask

b. [\*sudo\*] pip install flask-socketio

[\*sudo \*] only for Linux based systems.

**Basic Information:**

The application is made with Flask framework and Flask-Socketio package at the backend.

On the frontend it uses Bootstrap, JQuery, Jquery-cookie,Socketio libraries.

For simplicity purposes the application’s working and views are limited to the assignment’s

functionality. Hence, I have made only three screens.

**Working**

The Flask-Socketio package handles the socket connection between client and server while app

runs on Flask serving webpages based on the request received.

The paths in the app are as follows:

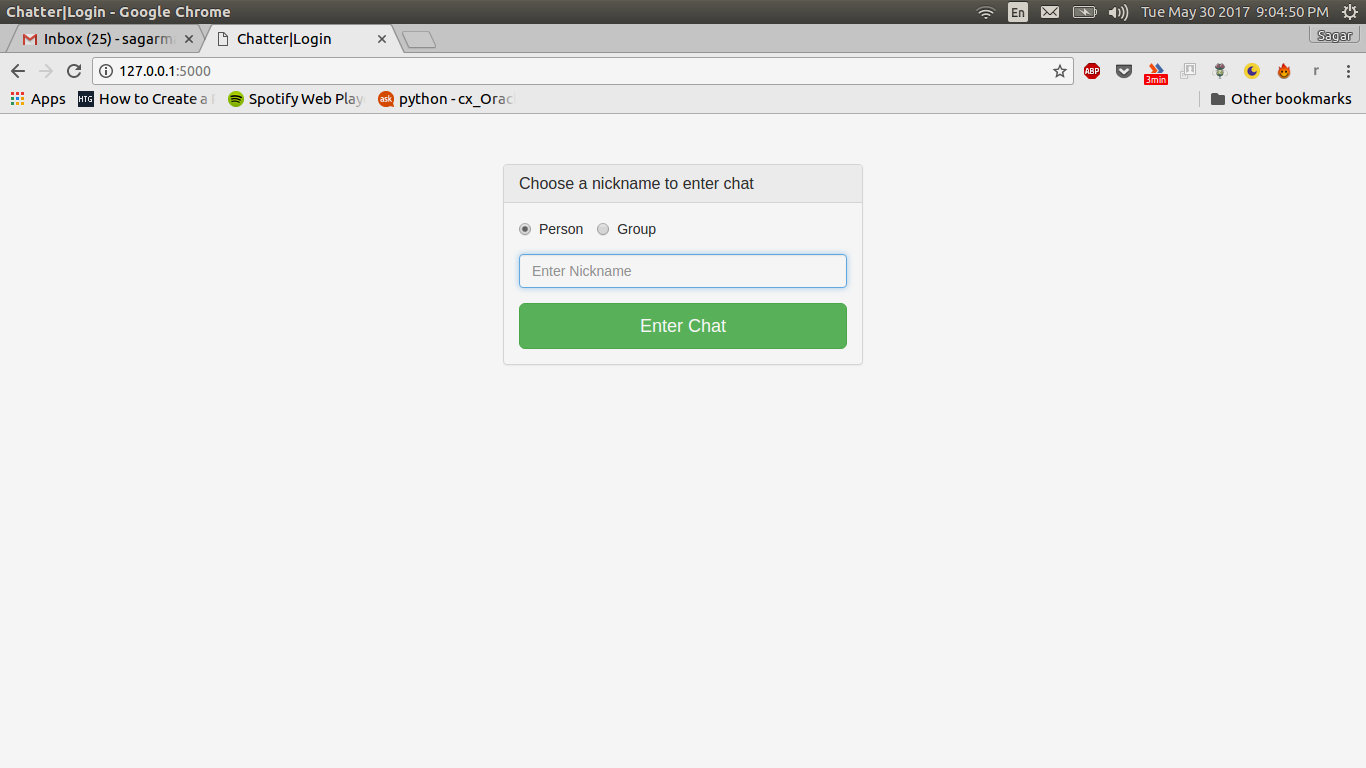
1. [**http://127.0.0.1:5000/**](http://127.0.0.1:5000/)**and**[**http://127.0.0.1:5000/login**](http://127.0.0.1:5000/login)

2. [**http://127.0.0.1:5000/person**](http://127.0.0.1:5000/person)

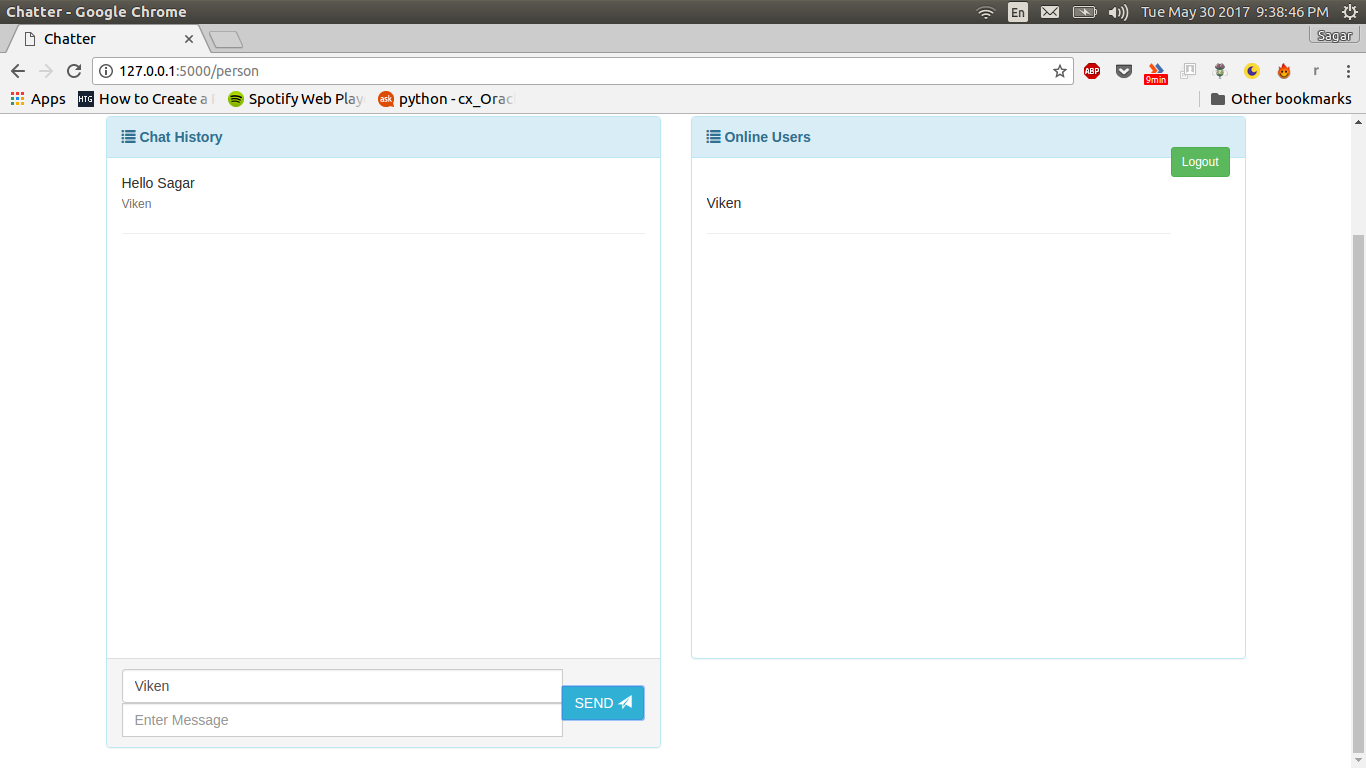
3. [**http://127.0.0.1:5000/group**](http://127.0.0.1:5000/group)

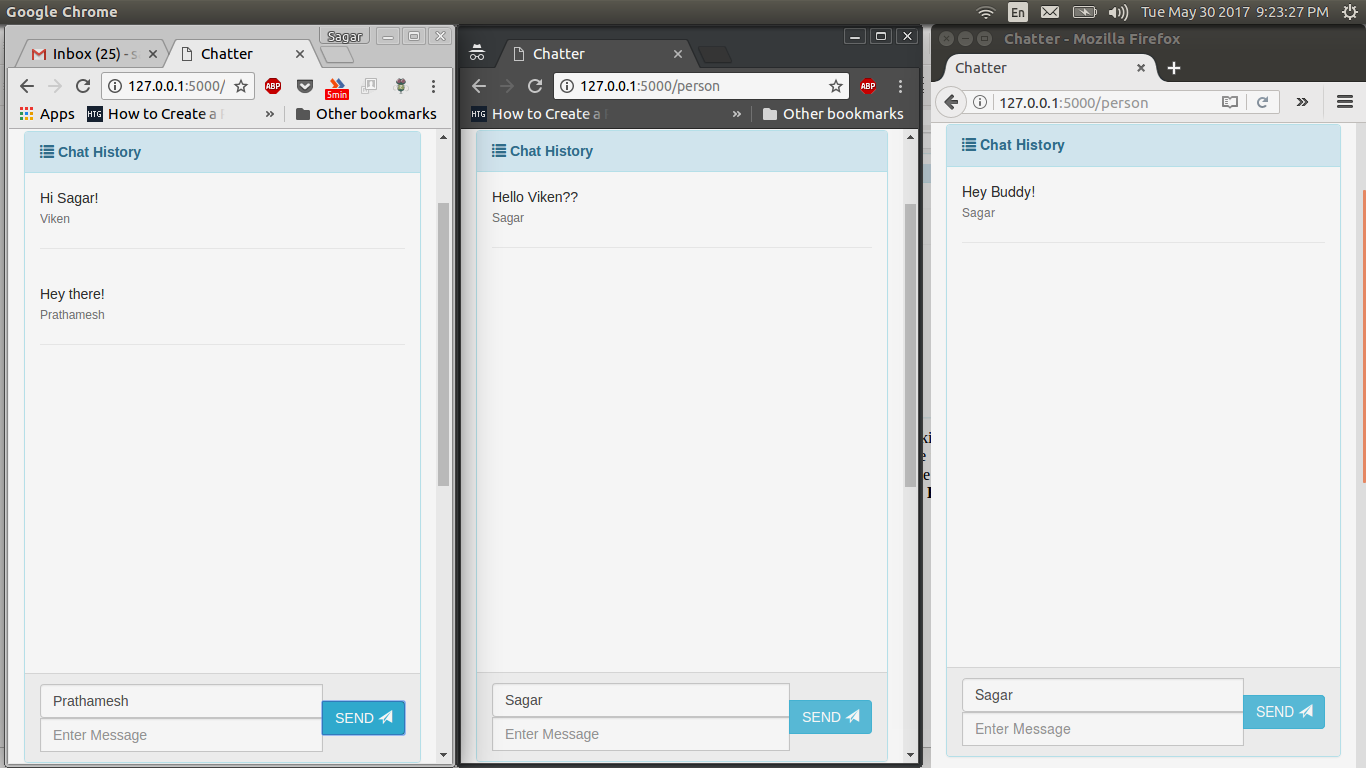
4. [**http://127.0.0.1:5000/logout**](http://127.0.0.1:5000/logout) **[\*Invoked only on logout\*]**

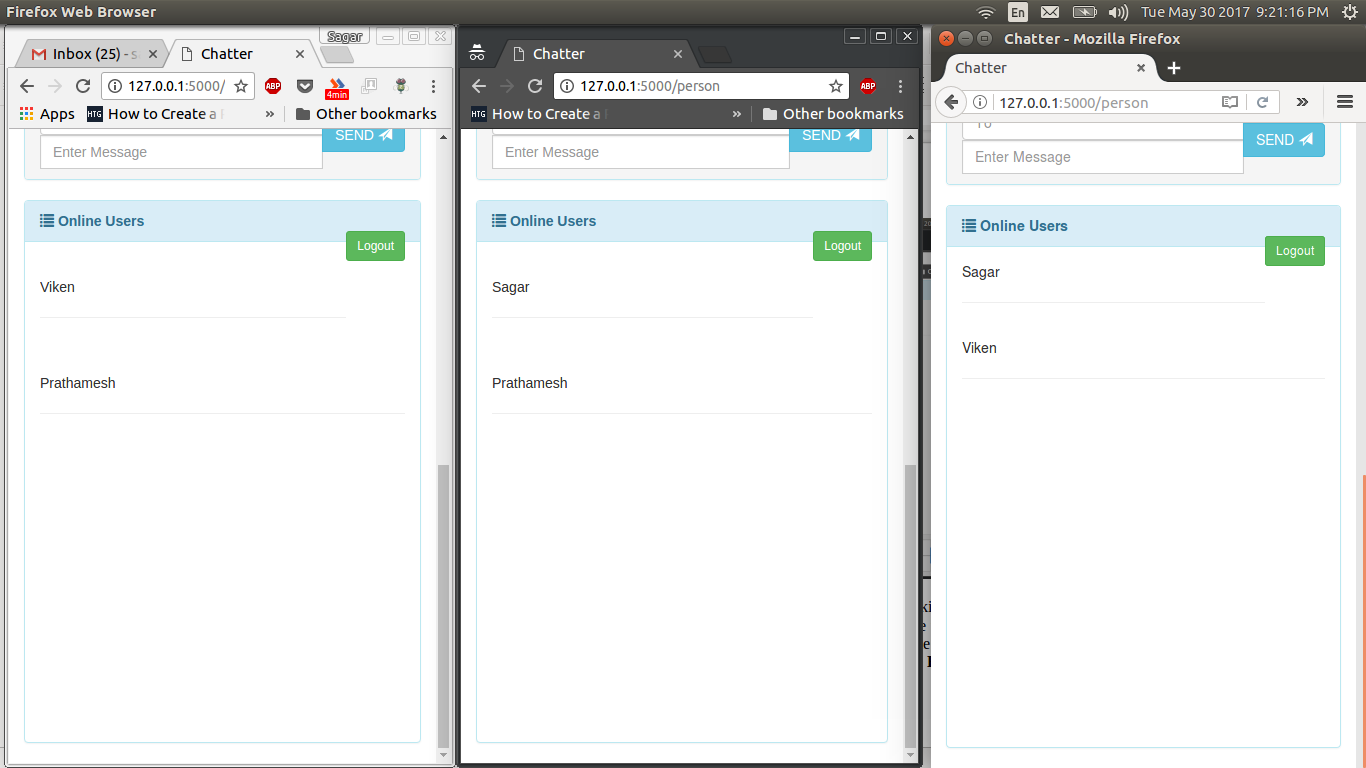
The working on the backend is in 3 parts:



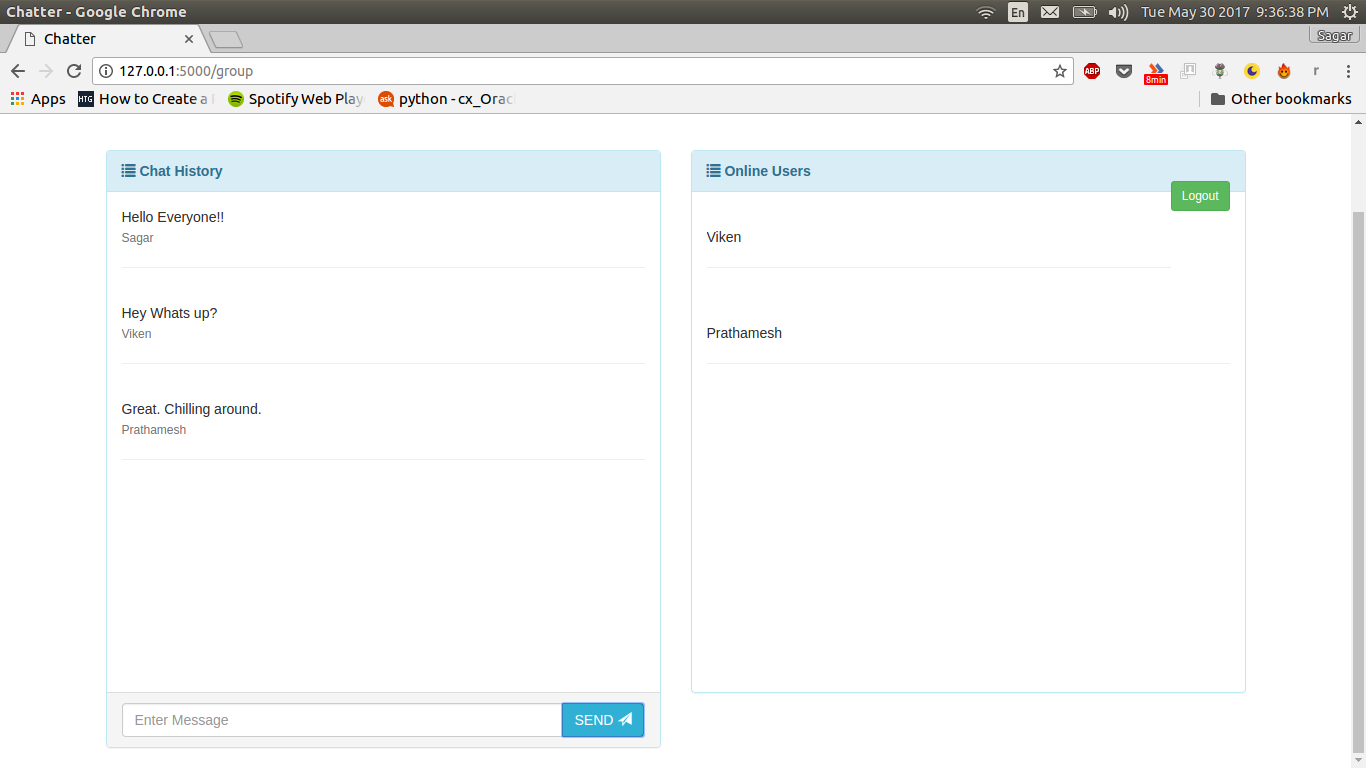
1. As we can see the first path is **‘/’ or ‘/login’**. This path is invoked when the user logs out and after logging out he/she is redirected to login page. Also if the user is trying to access the **‘/person’ or ‘/group’** and the cookie does not contains nickname then he/she is redirected to login page.

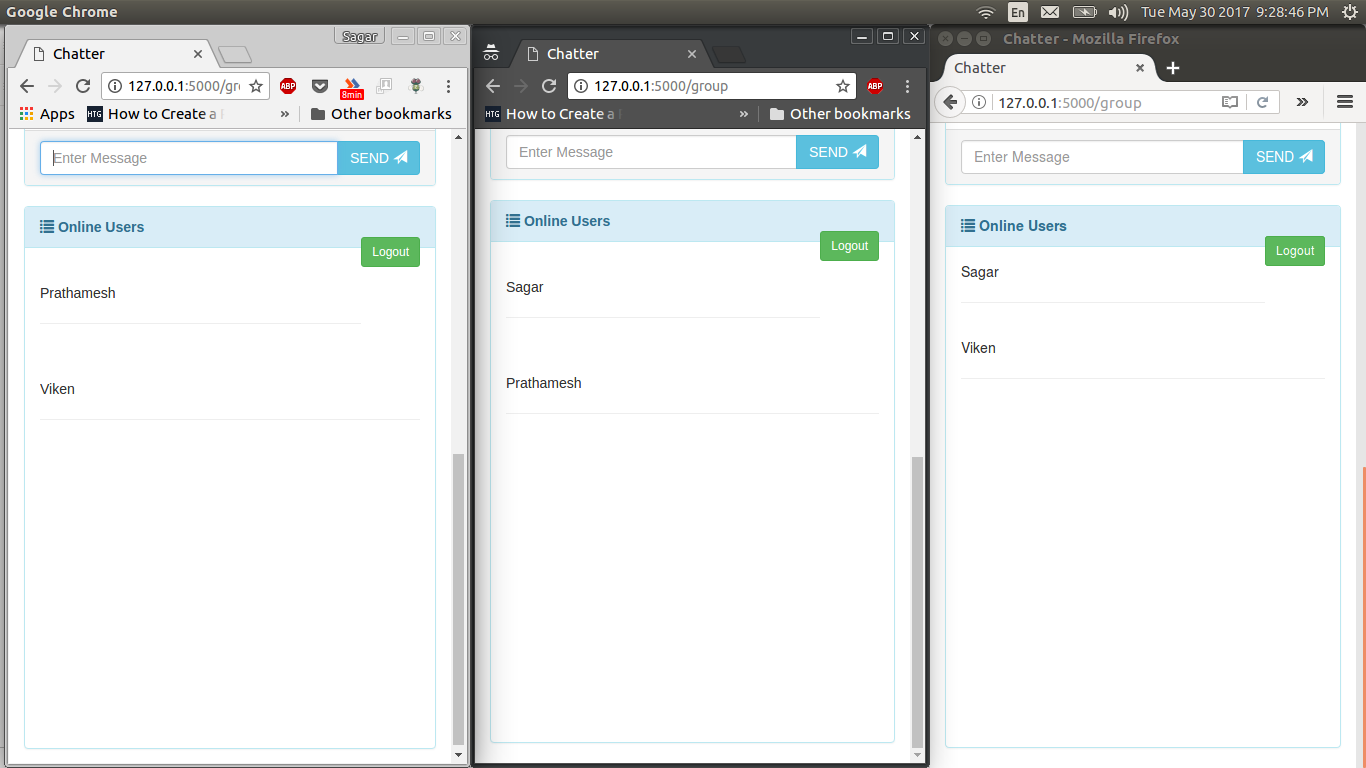


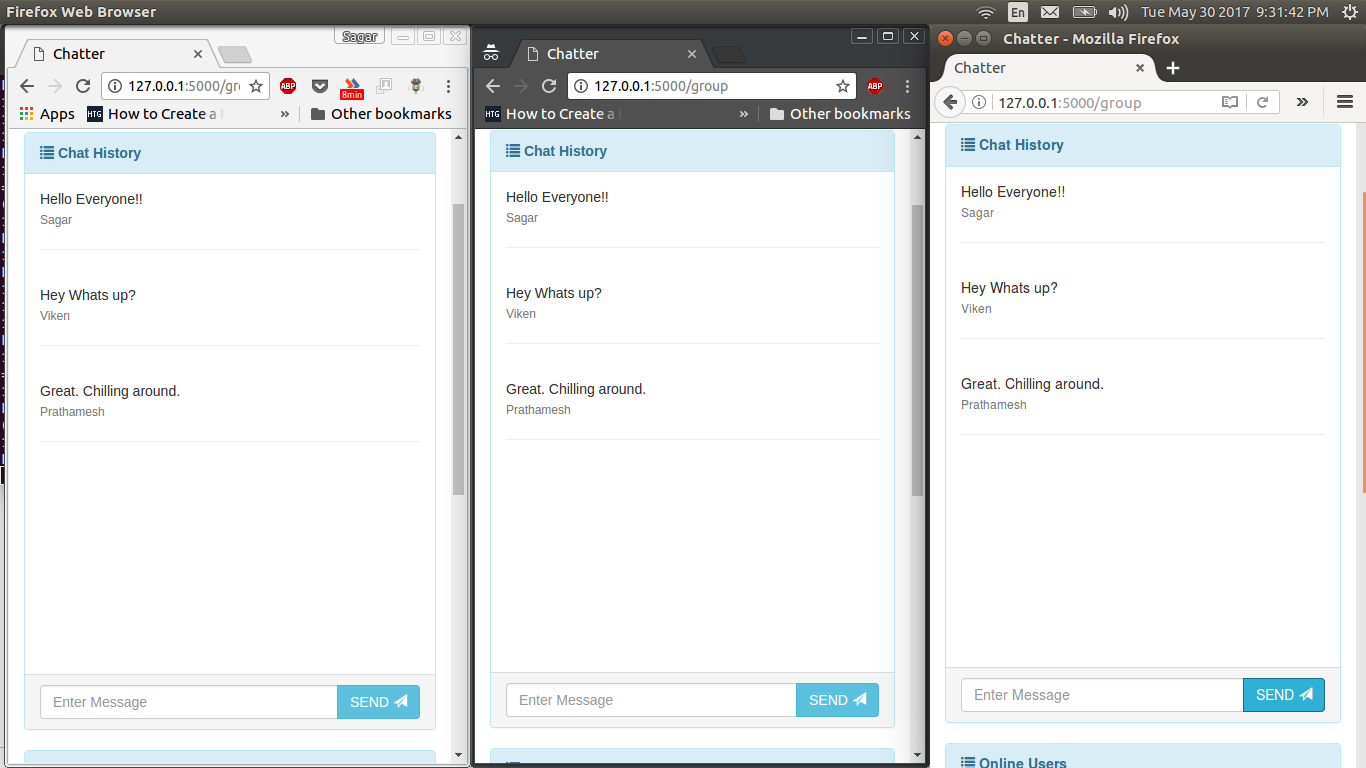




2. After entering the chat user id and selecting person option, the chat id is set in the cookie for usage in the application. The above screen is shown. The right hand side shows the online users who are also in person chat mode i.e. even they have selected the person chat option while logging in. For sending the message, type the person’s name in the **To** box and the message in the **Enter Message** box. The name can only be of that available in the online users list. The above screenshot explains how the application is working in person mode.







3. After entering the chat user id and selecting group option, the chat id is set in the cookie for usage in the application. The above screen is shown. The right hand side shows the online users who are also in group chat mode i.e. even they have selected the group chat option while logging in. For sending the message, type the message in the **Enter Message** box. The message is then sent to all the users in the group mode and online. The above screenshot explains how the application is working in person mode.

4. When the user clicks on **‘Logout’** button the user is logged out and the also the person disappears from the online users list. And he/she is redirected to **‘/login’** page.