Datanettverk og Skytjenester

DATA-2410 - Spring 2021

Obligatory Assignment 2

Group 61
Nima Abdollahi (s341890)
Glaysa Fernandez (s344047)
Ali Reza (s341823)

Starting the Program:

1. Libraries

To successfully run the program the following libraries must be installed:

```
$ pip install Flask
$ pip install restful-flask
$ pip install flask-login
$ pip install flask-socketio
$ pip install eventlet
```

or run requirements.txt file with the following command:

```
$ pip install -r requirement.txt
```

2. Running the server:

When using an IDE, simply click the run icon.

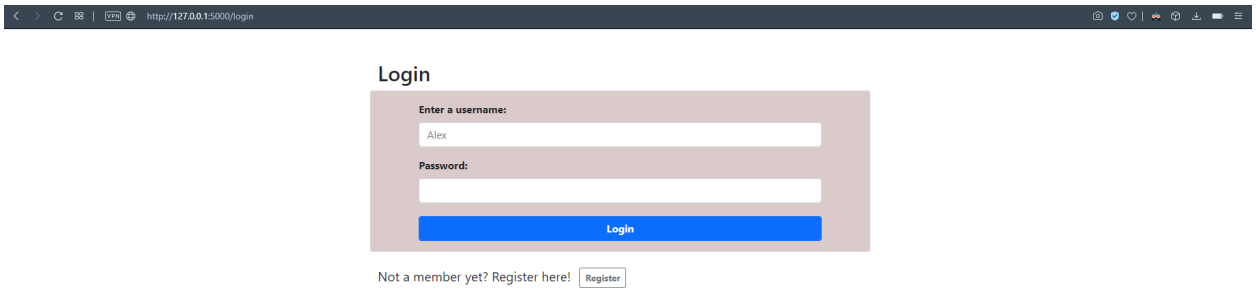
(PyCharm IDE)

When running the script on a CLI, you must always navigate first to the project's folder and type the following

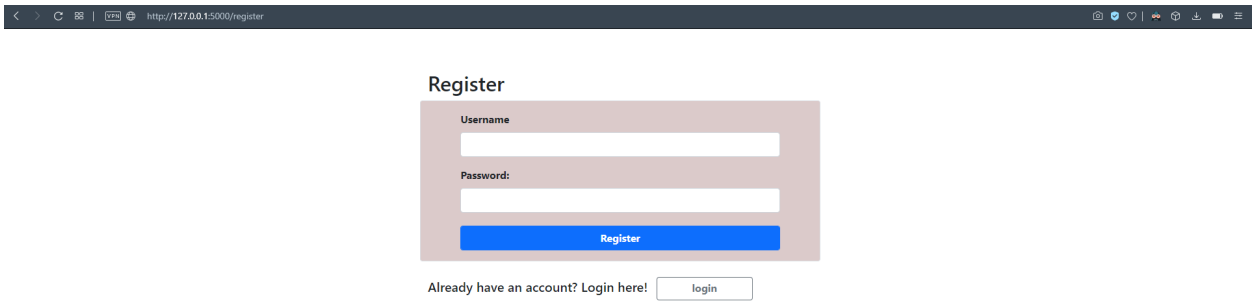
- Windows terminal: `start exe.bat`
- Unix terminal: `./exe.sh`
- Powershell: `.\exe.ps1`

3. Client

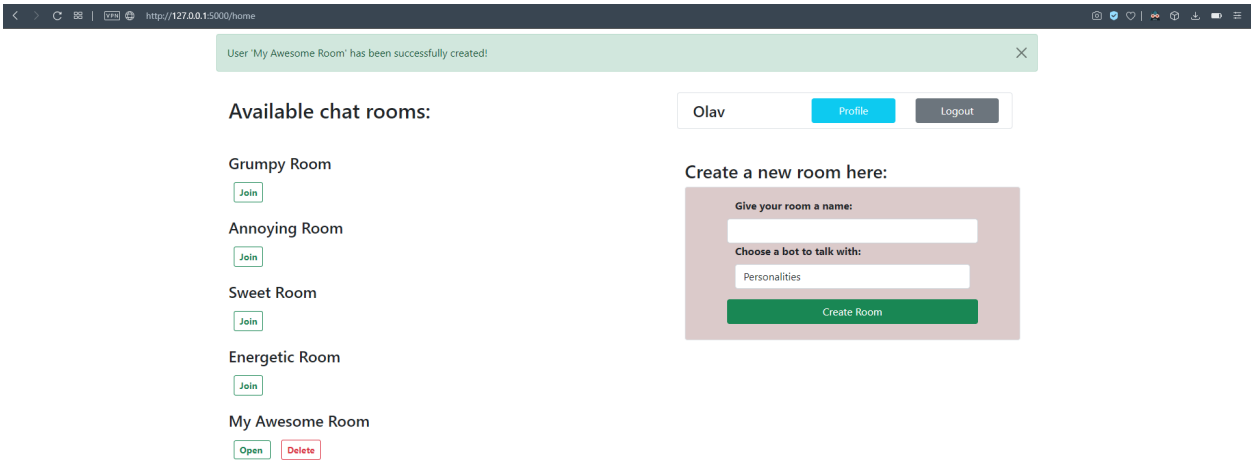
To access the client side, open up a browser and go to http://127.0.0.1:5000/. You will be prompted to login.



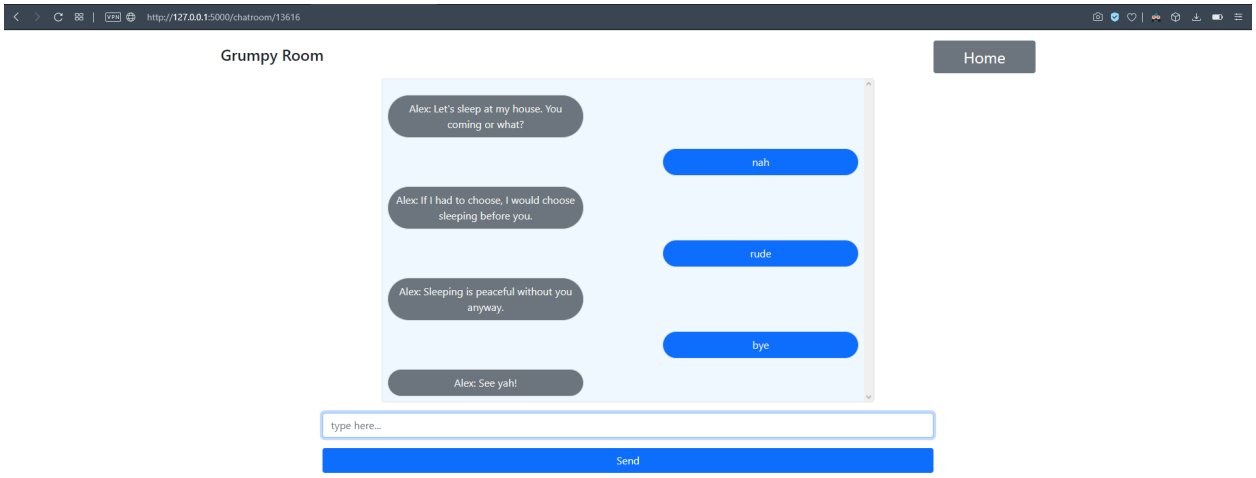
There are no registered users yet, therefore you must first register then login. You will be asked to enter a username and a password.



Once logged in, you can create and join chat rooms. You can only delete the rooms you have created. There are 4 already premade rooms you can join. The creator of these rooms are the bots. These bots will be a member of this room in which they can reply to whatever message you send. The rooms are named after their personalities. Whenever you create a room, a bot will automatically be a part of the room depending on which personality you choose.



Upon entering a chat room, a bot will automatically send a message and whenever you send a message, they will always reply.



You are also allowed to delete your own account.

Documentation:

1) Tools and environment

The web app is implemented using Jinja template and bootstrap for the client side. And, for the server side we have used flask.
The socket is implemented using Javascript for the client. And flask-socketio for the server.

2) REST-full API implementation

Our API provides access to different resources. We have in total seven resources. These resources are classes that inherit from the class `Resource` imported from the `flask_restful` package.With this class, we were able to implement supported HTTP methods such as get and post.

`Post` and `get` requests run in the background of the client interface. `Post` requests are called when registering and when the user sends a message to a chatroom. `Get` requests run when displaying all the rooms created by all users and when displaying all messages in a chatroom.

See the different resources by visiting http://127.0.0.1:5000 followed by one of the api endpoints below.

API Endpoints

```
"/api/users"
"/api/rooms"
"/api/user/<string:user_id>"
"/api/room/<string:room_id>"
"/api/room/<string:room_id>/users"
"/api/room/<string:room_id>/messages"
"/api/room/<string:room_id>/<string:user_id>/messages"
```

**PS: The id’s are hexadecimal numbers of 5 digits

3) Socket implementation

For the chatting functionality and transmitting data between users in a room, we have used `flask-socketio`. It makes the code both simpler and much efficient. The socket implementation code is divided into a client and a server part. The client part is implemented in javascript using JQuery. And, the server part is implemented in python.

Socket Server:

Path to code: `app/socket_views.py` and `app/__init__.py`
On the server first we begin by instantiating a SocketIO object by passing the Flask instance in it. And it establishes the socket server connection for the app.
Next we have events which are coming from the client, those are handled in `app/socket_views.py` file.

The first event handler is the `"join_room"` event. This event handler creates a room and adds the user to it. Then, It emits an event back to the client with data.

The second event handler is the `"send_message"` event. This event handler is responsible for sending the messages. It is doing that by assigning the right bot based on the data that it gets. The bot function takes the responsibility of instantiating a new message object. In addition to sending messages, this function postes new messages to the room's messages list.

Socket Client:

Path to code: `app/templates/chatroom.html`
The client part begins by creating a socket.io connection on the document domain and the location port. Later this connection object emits and receives events from the server. The chat begins with connecting the user that has entered the chatroom, then it emits the `"join_room"` event with a data as json which contains information of the room and the user.
After the connection event comes the `"send_message"` event. This event sends data in json format to the server `"send_message"` handler.
The last event on the client is the `"recive_message"` event. This one is responsible for getting the json data and displaying the content on the UI.