Datanettverk og skytjenester
18.04.2021

Obligatory assignment 2: wRESTling bots - Description

We started the assignment, dividing the work in two, on group takes task 1 and the other group takes task 2. Within the group of task 2, the group decided to also divide the task in half. One person takes part 1 of the task and the other person takes part 2. We have not been able to do the bots part of task 2 due to some miscommunication within the group.

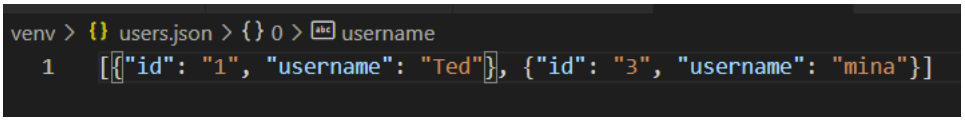
Used programming language: python 3

Description for Task 1: server.py

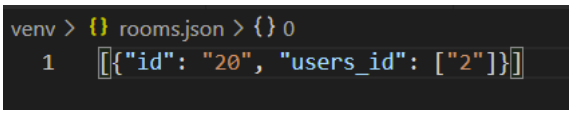
For this implementation we made class (Resource) for each route and then added each of those routes with api.add_resource(classname, route). We have 3 json files: messages, users and rooms. We will give you an example (screenshots) with some info inside the json files, and the expected output.

Starting off with the dicts like this:

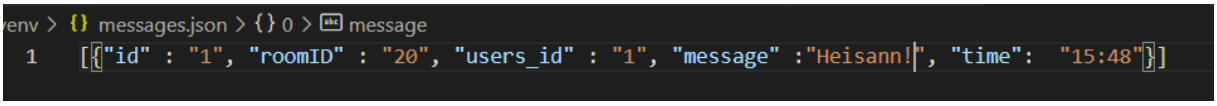
users.json



rooms.json



messages.json



Datanettverk og skytjenester
18.04.2021

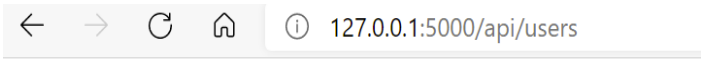
We run the server file in terminal like this:

```
(venv) PS C:\Users\sevde\Data-Oblig2> cd C:\Users\sevde\Data-Oblig2\venv\
(venv) PS C:\Users\sevde\Data-Oblig2\venv> python server.py
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 116-053-242
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Client file like this:

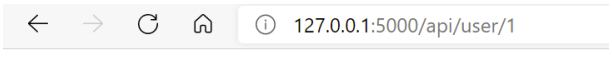
```
(venv) PS C:\Users\sevde\Data-Oblig2> & c:/Users/sevde/Data-Oblig2/venv/Scripts/python.exe c:/Users
/sevde/Data-Oblig2/venv/client2.py
[{'id': '1', 'username': 'Ted'}, {'id': '3', 'username': 'mina'}]
[{'id': '1', 'username': 'Ted'}, {'id': '3', 'username': 'mina'}, {'id': '2', 'username': 'Bob'}]
{'msg': 'deleted user'}
{'id': '1', 'username': 'Ted'}
[{'id': '20', 'users_id': ['2']}, {'id': '22', 'users_id': ['1']}]
{'erorr': 'roomID is taken'}
{'id': '20', 'users_id': ['2']}
{'users_id': ['2']}
{'msg': 'user exists in room'}
[{'id': '1', 'roomID': '20', 'users_id': '1', 'message': 'Heisann!', 'time': '15:48'}]
{'Data': [{'id': '1', 'roomID': '20', 'users_id': '1', 'message': 'Heisann!', 'time': '15:48'}]}
{'id': '2', 'roomID': '22', 'users_id': '1', 'message': 'sender meld funker', 'time': '2021-04-18 2
2:57:49.218922'}
```

Output in browser:



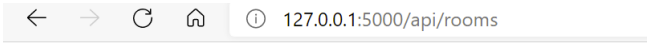

```
[
    {
        "id": "1",
        "username": "Ted"
    },
    {
        "id": "3",
        "username": "mina"
    }
]
```


We added a new user with id 2 but also deleted the user with id 2 so the remaining users are the ones in the picture above.

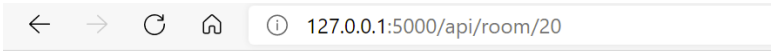



```
{
    "id": "1",
    "username": "Ted"
}
```

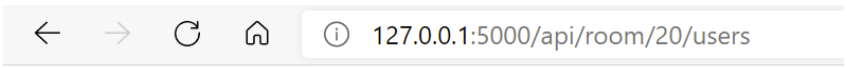
Datanettverk og skytjenester
18.04.2021




```
[
    {
        "id": "20",
        "users_id": [
            "2"
        ]
    },
    {
        "id": "22",
        "users_id": [
            "1"
        ]
    }
]
```

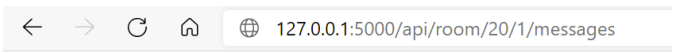

```
{
    "id": "20",
    "users_id": [
        "2"
    ]
}
```


```
{
    "users_id": [
        "2"
    ]
}
```



```
[
    {
        "id": "1",
        "roomID": "20",
        "users_id": "1",
        "message": "Heisann!",
        "time": "15:48"
    }
]
```

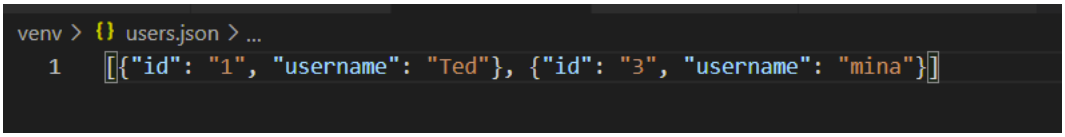




```
{
    "Data": [
        {
            "id": "1",
            "roomID": "20",
            "users_id": "1",
            "message": "Heisann!",
            "time": "15:48"
        },
        {
            "id": "2",
            "roomID": "22",
            "users_id": "1",
            "message": "sender meld funker",
            "time": "2021-04-18 22:57:49.218922"
        }
    ]
}
```

Datanettverk og skytjenester
18.04.2021

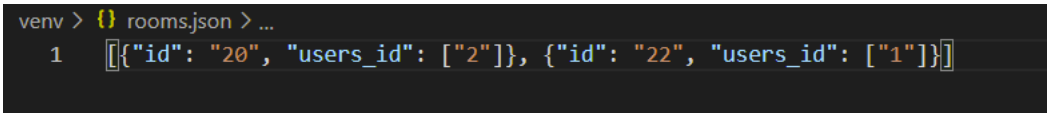
New json files after operations:

users.json



Still the same in users because we deleted the same user we added so we remain with the old json file

rooms.json



messages.json

Description for Task 2: client.py

For the implementation of task 2 part 1, we choose to use the HTTPS methods, mainly we used post-request and get request. We used post request to add new users, add chatroom ids and messages. We used get-request to get overview of users, rooms and to retrieve messages.

This is how you would register/add and new user, room or send a text message.

```
send = {"userID": "3","username":"mina"}
response = requests.post(url + "/api/users", json=send)
if response.status_code == 200:
    print(response.json())
else:
    print(response.status_code)
```

By writing the name, and the id that you want inside the json file. It's the same concept for the other post methods too, just write the needed information and send.

The program runs using the terminal, and also you can go to the web using the specific routs, for example : url /api/users as we showed in task 1 description