Lab Assignment 5: Writing a web service

In this assignment you will expand on the small web service we saw in class. In particular you will have to read and understand existing code, and extend/adjust it to meet your goals.

This assignment is more complicated. In particular, we will also be introduced to maintaining a project via git. You should start by cloning the project via the following steps:

- Open up the terminal to the location you want to work on. We will automatically created a subfolder in there.
- Do `git clone https://github.com/skiadas/messaging-flask.git`. This should download the whole project folder and create a folder called `messaging-flask` right where you are.
- Go into that folder via `cd messaging-flask`.
- You can now open the whole folder in SublimeText via `subl .`. There are a number of files in this folder that you need to be acquainted with.
- First you will find a `specs` folder with three documentation files. We have seen some of those files in class already. They describe the project in some detail. You will in particular need to study the `resources.md` file that contains information about the different routes and their expected results.
- The files you will work on are all in the `app` folder.
- You will want to have GitKraken open on this project. When you have completed a solution for one of the problems, you want to review the changes in GitKraken, and then stage them and create a commit. The message of the commit should include the number of the assignment problem that the changes in this commit address.
- In order to use the server, you will need to tell it about your database. You will do so by creating a file called `keys.json` in the `app` folder. It should look like this: `json { "USERNAME": "username here", "PASSWORD": "database password here", "SERVER": "vault.han`
- You will need to have two terminals open, both at the `messaging-flask` folder. The one terminal will run a "server". You start that server with the command `python app/messaging.py`. You periodically will want to shut the server down with Ctrl-C and then restart it.
- The other terminal is your "client" that you can use to test your server. You will start a python session with `python3`, then run `import requests` to load a very popular requests package. Now you can perform requests with commands like `r = requests.get('http://127.0.0.1:5000/users/haris')`.
- In the `app` folder are the files you will need to work on and add your code. You will want to study those files and try to understand what they are doing right now.

- messaging.py is the main controller file that kickstarts everything. It starts a "Flask" application, sets up some database information, and then contains a list of the various routes, along with the methods that handle them. When you need to implement a new route, or change the fundamental behavior of a new route, you will need to work on this file.
- message.py is a helper file that is meant to contain methods useful when processing messages. You can access it from within messaging.py via the message variable. If you need to do some validation of your inputs, or some other work related to your data, this is a good place to put that.
- db.py is a helper file that implements the various database interactions. You will be writing some sqlalchemy code in here. Whenever you need to get something from or write something to the database, you want to write the corresponding function here, and access it from messaging.py via the db variable prefix.

Specific problems follow. You should only attempt these after you have a good understanding of how the existing code in the file works.

1.

When you are ready to submit, make sure you have all your changes saved via commits, then run the following:

```
git format-patch v1..master --stdout > yourName-assignment5.patch
```

Of course use your name in the filename at the en of that line. This will create this new file, which contains all the information about the change you have done. Email this file to me in order to submit.