Introduction of Network Programming 2018 Autumn

Homework 3 - Social Service: Server

Description

In this assignment, you are going to create a server side program of a social service. Your program should be able to handle request from your homework 2 program, and return the correct response.

Requirement

The service accept the following commands:

Request Format	Description	Return to client (in JSON format)	

register <*id*> <*password*>
 	
done
	Register with id and password.
	● <id> must be unique.
	● Password has no limitation.
		Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “<*id*> is already used”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: register <*id*> <*password*>”
			}

login <*id*> <*password*>
 	
done
	Login with the id and password, and return an access token. If there’s already an access token exist, you should just return that one.
	● Access token must be unique.
		Success	{
			“status”: 0,
			“token”: “11d28a07-2091-4c57-ba46-7fb1bc488cd6”,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “No such user or password error”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: login <*id*> <*password*>”
			}

delete <*token*>
 	
done
	Delete a user account. All the relative data will be deleted, such as posts, friends and tokens.	Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: delete <*user*>”
			}

logout <*token*>
 	
done
	Logout the user (Invalidate the current access token)	Success	{
			“status”: 0,
			“message”: “Bye!”
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,

			“message”: “Usage: logout <*user*>” }

invite <*token*> <*id*>
 	
done
	Invite <*id*> to become your friend.
	● If <*id*> has already invited you, you are not able to invite <*id*>.
	● You cannot invite yourself.
		Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “<*id*> is already your friend”
			}
			or
			{
			“status”: 1,
			“message”: “<*id*> does not exist” done
			}
			or
			{
			“status”: 1,
			“message”: “Not login yet” done
			}
			or
			{
			“status”: 1,
			“message”: “You cannot invite yourself” done
			}
			or
			{
			“status”: 1,
			“message”: “Already invited”
			}
			or
			{
			“status”: 1,
			“message”: “<id> has invited you”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: invite <*user*> <*id*>” done
			}

list-invite <*token*>	List all the users who invited you to become friends.	Success	{
			“status”: 0,
			“invite”: [
			user_A,
			user_B
			]
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: list-invite <*user*>”
			}

accept-invite <*token*> <*id*>	Accept the <*id*>’s friend invitation.	Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “<*id*> did not invite you”
			}
			or
			{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{

			“status”: 1, “message”: “Usage: accept-invite <*user*> <*id*>” }

list-friend <*token*>	List all your friends.	Success	{
			“status”: 0,
			“friend”: [
			user_A,
			user_B
			]
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: list-friend <*user*>”
			}

post <*token*> <*message*>	Share a post with your friends
	● <*message*> accept spaces
		Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: post <*user*> <*message*>”
			}

receive-post <*token*>	Receive the posts from your friends.	Success	{
			“status”: 0,
			“post”: [
			{
			“id”: user_A,
			“message”: “I have no friends”
			},
			{
			“id”: user_B,
			“message”: “I have no friends too”
			}
			]
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: receive-post <*user*>”
			}

General

- In order to serve the clients, your server program **must** accept **two command line arguments: ip and port**.
- Your program have to handle the requests above and reply the appropriate response. Therefore, you may have to manage a **database** to store the data. Please design the tables by yourself.
- For the commands that contain <*token*> field, your server should first validate the token, and manage the resources based on token owner.
- The response format **must** be JSON format.
- If online users delete their accounts, you **must** also logout the account.
- The message in **post command allows spaces**.
- **Please note that you will need to serve more commands in homework 4. Take care about the extensibility and readability.**

Sample Input

```
register testA 111
register testB 222
register testC 333
login testA 111
login testB 222
login testC 333
invite testA testB
accept-invite testB testA
invite testB testC
accept-invite testC testB
list-friend testA
list-friend testB
list-friend testC
post testA Hi I am A
post testB Hi I am B
post testC Hi I am C
receive-post testA
receive-post testB
receive-post testC
delete testA
delete testB
delete testC
exit
```

Sample Output

```
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
testB
testA
testC
testB
Success!
Success!
Success!
testB: Hi I am B
testA: Hi I am A
testC: Hi I am C
testB: Hi I am B
Success!
Success!
Success!
```

Grade (100%)

- Able to open a service through command line arguments - (5%)
- Able to deal with unknown command and arguments - (5%)
- Each command: 9%, 10 commands in total - (90%)

Submit

Please upload a zip file called “hw3_{$student_id}.zip” that includes your source code. Submission that don’t follow the rule will get 20% punishment on the grade.

Demo time will be announced before the deadline. You are not allowed to modify your code after demo. Please submit your code on time.

If you have any questions, please ask your questions on course forum(https://e3new.nctu.edu.tw/)

Reference

1. **JSON format (https://zh.wikipedia.org/wiki/JSON)**
2. **C socket (http://man7.org/linux/man-pages/man2/socket.2.html)**
3. **Python socket (https://docs.python.org/3/library/socket.html)**
4. **C JSON (https://github.com/json-c/json-c)**
5. **Python JSON (https://docs.python.org/3/library/json.html)**
6. **C++ ODB (https://www.codesynthesis.com/products/odb/)**
7. **Python peewee (http://docs.peewee-orm.com/en/latest/)**