Third Assignment

Q1

We extended the test server to accept multiple connections at the same time using multi-threading, also we are using now users `sqlite` database instead of files. The database contain one table `users` with tow columns `username` and `password` we are still storing the passwords using `hash` functions for extra security, we implemented three functions to deal with the database `create_db` creates the database if it's not created yet, `insert_user(username, password)` inserts new user to the database and `query_user(username)` to check if some user is already stored in the database.

When a new connection comes to the server we are creating a new `ClientThread` the `__init__` function is:

```
def __init__(self, ip, port, count):
    Thread.__init__(self)
    self.ip = ip
    self.port = port
    self.count = count
    self.name = 'Thread - ' + str(count)
    print "[+] New server socket thread
     started for " + ip + ":" + str(port)
```

Also we have overridden the `run` function, when the `thread` starts, first we check if this is the seventh connection, if yes we call the `send_threads` function, this function sends all the connections ports and IPs, the output of this function looks at the client like this:

```
Congratulations you are the sevnth user!!!
```

```
127.0.0.1:38184
127.0.0.1:38188
127.0.0.1:38190
127.0.0.1:38192
127.0.0.1:38194
127.0.0.1:38196
127.0.0.1:38198
```

The rest of the `run` function is:

```
username = authinticate(client_socket)
do_exam(client_socket, username)
print('closing socket for user ' + username)
client_socket.close()
```

The server still accepts multiple users, when a new user registers he is registered to the database, generally everything else is same as the previous assignment.
P.S: We are sorry for being late to deliver the assignment, but we were trying to make a `GUI` for the client using `tkinter` but we had many bugs and couldn't make it in time, however it should be ready before the interview time.

Q2

First we have imported all the required libraries

```
from flask import Flask,render_template,
redirect,url_for,request
import sqlite3
```

Then we have created a new `sqlite` database called (Student_log) using function `con_db` , after connecting the app with database and creating a new cursor, we have created a table (Fifth_students) contains three columns `Name` , `ID` and `Age` .
After that we have defined 4 functions to control our database and it's table which are:

- insert_student(S_Name, S_ID, S_Age)` used to insert a new student's data.

- `clear_students()` used to delete the whole content of the table.
- `selectall_student()` used to fetch the whole content of the table
- `selectone_student(S_ID)` to fetch specific data using conditions (the student with the exact entered ID).

  Then we created a new `flask` app that have three routes `home, login and handlelogin`

  Each route has a function:
- `home()` redirects the user to login page which is a html file `login.html` in templates folder
- `Login()` renders the login template which has a form to enter the required ID and send it to `handlelogin` function using `POST http` request
- `handlelogin()` takes the entered ID so we process two situations:
  - the ID exits in the database so the function fetches it as list and slices it and sends the data to handlelogin template `handlelogin.html` then renders it.
  - the ID does not exist in the database so the function renders error template.

When we start the server using the command:

```
Flask_Web.py
```

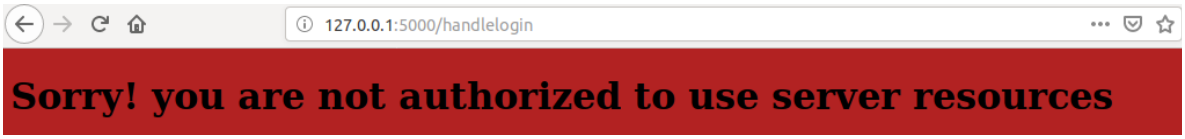
We get something like:

```
* Serving Flask app "Flask_Web" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/
* (Press CTRL+C to quit)
```

The login page looks like this:

After successful login :

Unsuccessful login:



Server status:

```
(venv) ali-ibrahim@C137:~/GitHub/Network-programming-Assignments/Third Assignment/Q2 Flask app$ python Flask_Web.py
 * Serving Flask app "Flask_Web" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/May/2019 04:29:28] "GET / HTTP/1.1" 302 -
127.0.0.1 - - [27/May/2019 04:29:28] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2019 04:29:29] "GET /favicon.ico HTTP/1.1" 404 -
[('Sara', 1800, 23)]
127.0.0.1 - - [27/May/2019 04:34:37] "POST /handlelogin HTTP/1.1" 200 -
[('Ali', 1866, 23)]
127.0.0.1 - - [27/May/2019 04:34:43] "POST /handlelogin HTTP/1.1" 200 -
[('Sara', 1800, 23)]
127.0.0.1 - - [27/May/2019 04:34:47] "POST /handlelogin HTTP/1.1" 200 -
[('Karam', 1711, 23)]
127.0.0.1 - - [27/May/2019 04:34:51] "POST /handlelogin HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2019 04:36:48] "POST /handlelogin HTTP/1.1" 200 -
```