**Facebook simulation application**

You are going to design a pseudo **console** web-application similar to Facebook. You can choose any name that you like for your application other than Facebook, in this case I will choose the name “Horizon”.

I: Part

The whole application is divided into two parts:

- The main application: Also known as “Horizon main”. This is the part where only users can interact.

- The system: Also known as “Horizon system”. This is the part where only admins can interact.

II: Roles

There will be two types of people who will use the application the one is the users and the one is the admins.

The users however is a little bit special. Every user can only use the application via this current machine! . Therefore you can create multiple accounts that represent different users. In general you can use the term users and accounts interchangably.

1) The users.

The users are those who use your Horizon main . Lets talk about one specific user since every user has the same result using your application.

When a user starts Horizon main.

- He will see the main page (**section A**) which:

Print “Welcome to Horizon” at the top of this section. Below the message print “Login(L), Register(R), Guest(G)” .Below that print “Where do you want to go?” which asks for user input . The input only accept values “L”, “R” or “G” if the user enter something else even the lowercase version of these letters. He will be asked to input again.

* If the user has successfully specified the value:
* inputted R: He will be headed to the Register section. Specifically after the input, the console will print Username, Email and Password which all require input. Here are the rules of the inputs:

|  |  |
| --- | --- |
| Username | A string whose length is from 5 to 20 any characters. |
| Email | Satisfies the format [XXX@YYY.com](mailto:XXX@YYY.com) where both XXX and YYY only contains alphanumeric characters. The “com” is case sensitive! |
| Password | Contains at least one uppercase and one lowercase. Especially DOESNOT contain case – insensitive sequential string whose length higher or equal than 4 such as “12345”,”56789”,”defghij” or “AbcDE” but “123” is ok .The length of the password should be at least 10. |

+ If he fails to input in one of these, he will be asked to input **the whole form** again.

+ If he successes to input all then after the password input, the inputted information will be saved in the csv file named **UserInfo.csv** (more about this csv later). After that he will be automatically headed back to section A.

Note: - The password must be **hashed** before storing into the csv file.

- The inputted email must be **unique** (Doesn’t identical to any emails in the csv).

Special rule: He can input “=>A” at the end of one of the required field in order to head back to section A and all inputted values will be lost.

* inputted L: He will be headed to the login section. Specifically after the input, the console will print Email and Password which both require input:

+If he fails to input in one of these, he will be asked to input **the whole form** again.

+If he inputs the correct combination of Email and Password which available in the csv rows. Especially the hashed inputted password must match with the hashed password in the csv. After the Password field, the user will be headed to the section B.

Special rule: He can input “=>A” at the end of one of the required field in order to head back to section A and all inputted values will be lost.

Important note: In this application, whenever the user has a chance to input a command like “=>A” that command will NOT be part of that entire input. You can support your own defined escape sequences as well or just assume the user will never input his real data that include a command. Moreover double quotes are NOT part of commands (I use them to highlight command syntax for this document only).

* inputted G: The console will print the content of **Guest.txt** file. After that the user will be headed back to section A.

- **Section B**:

This is the real part of Horizon main where the users will use the true features.

Important Note: It doesn’t matter when another user can see the previous content by scrolling up the console. We assume that the all previously printed content in the console will be invisible !

Log out: At any point in this section where the user can input. At the end of that input, if he input “=> L”, he will get logged out meaning that the user will be headed back to section A and this is the only way to get to section A in this section B (This command is not ambiguous with the Login one since now we are in section B)

Remember me:

The user can choose to be remembered by the application. When that happens:

(1) If the user doesn’t log out but close the Horizon main and then reopen it, the user will be headed right to section B (of that user).

(2) If the user has logged out already but has chosen to be remembered then after he close the Horizon main and then reopen it, he will be headed to section A and **Not** headed right to section B.

(3) After the (2) event, if the user re-login then event (1) and (2) are still apllied in section B. Meaning that the user **is still remembered** by the application.

If the user doesn’t choose to be remembered

The second time that user reopen the application, he will be headed to section A and **Not** headed right to section B.

*In conclusion: Only* ***one*** *user can be remembered at a time!*

The control of Remember me:

- In order to be remembered: input “=>R” at the end of the input whenever the user has the chance to input.

- In order to Not be remembered: input “=>C” (cancel) at the end of the input whenever the user has the chance to input. It he does so then it is equivilant to the user doesn’t choose to be remembered.

- If he use “=>R” then he cannot use this command at the **consecutive** second time and in that second time “=>R” will just be a normal string which is a part of user input. The same goes for “=>C”

- If he use “=>R” then “=>C” is enable and “=>R” is disable. Similarly, if he use “=>C” then “=>R” is enable and “=>C” is disable

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| =>R | =>C | =>R | =>C | =>R |

In conclusion: The two remember commands have to be used alternately like this array (position 0 must be “=>R”)

The array can go infinitely in this pattern.

About **UserInfo.csv**:

This csv file is used to store necessary user information for application functionality.

There should be at least 3 columns: Email, Username, Password. You can add more columns if you want . In regard to password, the csv **only** contains **hashed** password of the user.

For remember me: Up to you to design how you support remember me functionality. In here I would add one more column called Remembered which:

Only contains boolean values however there is only ONE account has the “true” value for Remembered field, the rest of the account will be set to false. There might be all users Remember field set to false. The “true” value denotes for **one** user is remembered in the application.