Deign Document: Manager Layer – User Info Manager

Hang Yuan ([hyuan211@gmail.com](mailto:hyuan211@gmail.com))

Version: 0.1 (09/02/19)

**1 Goals**

This module will manage the information of each user in both local client system and remote server system. Because client side and server side have different functionalities needs of user info management, some specific functions and file formats will be different. This design document will introduce all functions needed but the actual codes will have minute differences.

**2 Design**

The design for User Info Manager module includes two parts: (1) system design; (2) file type and format.

**2.1 System Design**

The module will manipulate the *user.sys* (on local client) or *user.data* (on remote server) to manage all the users’ basic information by reading, updating, creating or deleting user information or user account.

Notice that due to better organization, the *user.data* on server will be adapted to *domainName.data* with different domain names.

**2.1.1 Functions differ depending on sides**

Basic functions will be:

CreateUser (const string &userAccount) on client

CreateUser (const ServerUserInfo &userInfo) on server

RemoveUser (const string &userAccount) only on client

CloseUser (const string &userAccount) only on server

ReadUser (const string &userAccount, ClientUserInfo &userInfo) on client

ReadUser (const string &userAccount, ServerUserInfo &userInfo) on server

UpdateUser (const ServerUserInfo &userInfo) only on server

Login (const string &userAccount) on client

Login (const ServerUserInfo &userInfo) on server

Login (const string &userAccount)

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Aim** | |
| **Local client** | **Remote server** |
| CreateUser | Add the new user info into the user.sys file. Ignore if it exists. | Add the new user info into the specified *user.data* file |
| RemoveUser | Remove this user’s info from *user.sys* file | N/A |
| CloseUser | N/A | Delete relevant user information from the *user.data* file |
| ReadUser | Read and return all information of this user from its system file | |
| UpdateUser | N/A | Update relevant proportion of user information. If the user info doesn’t exist, create it first |
| Login | Update user log info | Verify account and password with the given information |
| Logout | Update user log info | Update logout time |

**2.1.2 UserInfo structure in different sides**

Based on different requirements of user info in client and server model, different structure will be adopted.

Under local client:

Struct ClientUserInfoHeader {

unsigned totalUserNumber;

};

Struct ClientUserInfo {

char userAccount[30]; // Example: user@example.com

};

Under remote server:

Struct ServerUserInfoHeader {

Unsigned totalUserNumber;

};

Struct ServerUserInfo {

char username[15]; // Example: user

char domainName[15]; // Example: example.com

char password[16]; // Example: 1a2b3c4d

time\_t lastLoginTime;

time\_t lastLogoutTime;

bool changeStatus; // True: changed, False: no change

};

**2.2 File Type and Format**

**2.2.1 File types**

Based on the different usage of storing user information, the local client adopts *.sys* as the file type and remote server adopts *.data* as the file type.

**2.2.2 File format**

Client:

user.sys file will be responsible to save all the user locally signed in. The system file will have a header to indicate the number of the users in this file.

Server:

Users will be organized by its