CS MINI IA

Contents

[Reference 3](#_Toc51591368)

# Rationale

In this project, I designed a multi user online message software.

# Requirements

* The user can use the program with GUI interface
* The user can sign up with a username and password and get a random ID
* The user can sign in with the username and password
* After sign in
  + The user can change password
  + The user can request to add friends with friend’s ID
  + The user can choose to accept or refuse friend requests
  + The user can send and receive text message to friends
  + When messages are sent to off-line users, they will be stored on cloud, and when the receiver get online, he or she can get message to local. Then the message will be stored in local and removed in cloud.
  + The user can choose to remove message history from local database.

# Plan

* Develop language: Python3
* GUI Lib: Tkinter
* Database: sqlite3
* Communicate protocol: TCP

# Design

## Communication design

Each time client send a message to server in dictionary.

It have key “mode”, server end respond according to “mode” key’ value.

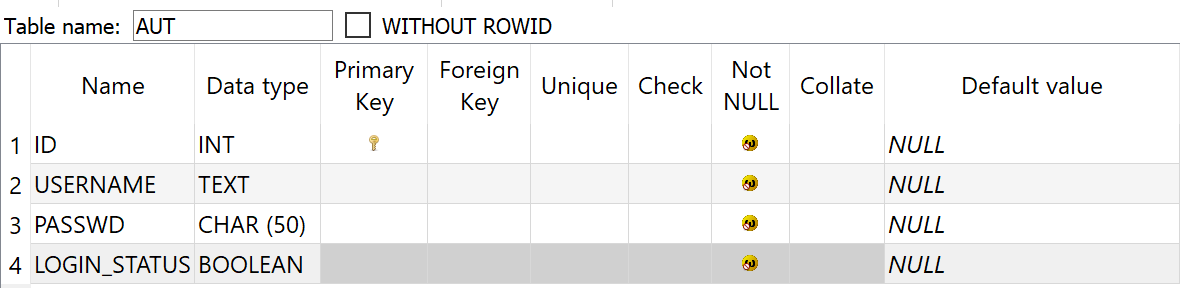
It have other keys and values needed for the mode

|  |  |  |
| --- | --- | --- |
| Mode | Keys | Server respond |
| “sign\_up” | “name”, “pwd” | Success: auto-assigned user\_id  Fail: 0 |
| “sign\_in” | “id”, “pwd” | Success: user\_name  Fail: 0 |
| “sign\_out” | None | Success: 1  Fail: 0 |
| “add\_friend” | “friend\_id”, “req\_note” | Success: 1  Fail: 0 |
| “del\_friend” | “friend\_id” | Success: 1  Fail: 0 |
| “accept\_friend” | “friend\_id” | Success: 1  Fail: 0 |
| “refuse\_friend” | “friend\_id” | Success: 1  Fail: 0 |
| “send\_msg” | “friend\_id”, “msg” | Success: 1  Fail: 0 |
| “refresh” | None | Success: {msg: [[sender\_id, time, msg], [], …], req: [], fl} |

## Database design

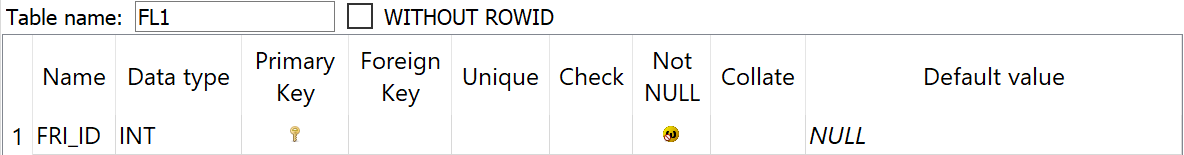
### Server end

One **AUT** table store the all users’ information

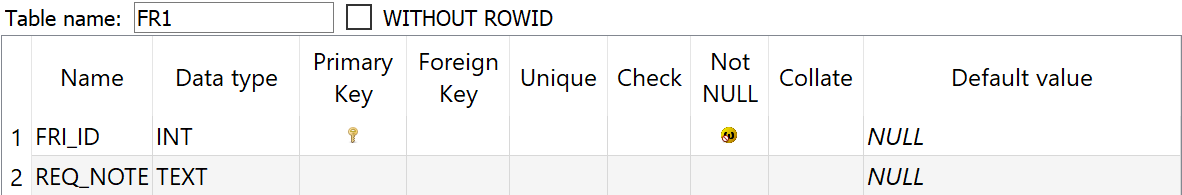


One **FL<ID>** table for each user store friends’ ID

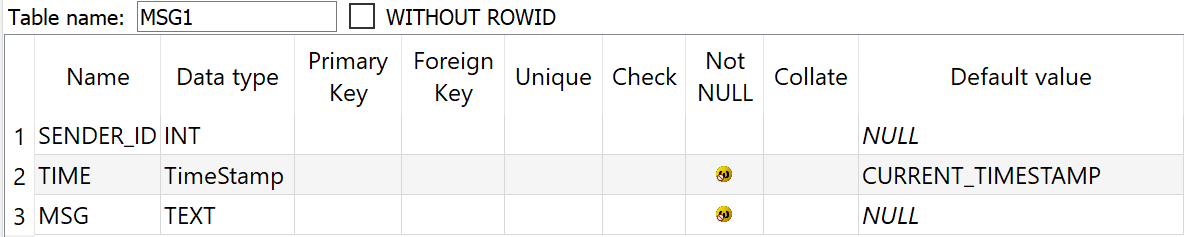
\*FL<ID> , for example ID is 114, the table name is FL114



One **FR<ID>** table for each user store friend requests info



One **MSG<ID>** table for each user store unreceived messages



### Client end

## UI design

## Top down design

# Develop

Code in appendix.

# Testing

# Deploy

# Maintains

# Reference

1. Eric Jin . multi-user-chatting-software . <https://github.com/EricEricEricJin/multi-user-chatting-software>
2. Eric Jin . Python DIY简易多人聊天软件 . <https://blog.csdn.net/weixin_43528943/article/details/104721649>
3. Runoob . SQLite – Python . <https://www.runoob.com/sqlite/sqlite-python.html>