目录

[1 引言 2](#_Toc54121658)

[2 项目设计 2](#_Toc54121659)

[3 详细设计 6](#_Toc54121660)

[4 实现方案 8](#_Toc54121661)

[5 系统测试 15](#_Toc54121662)

# 1 引言

1.1项目概述

项目目的：在已掌握的数据结构和算法的相关知识基础上，借助数据库和Qt以团队形式开发个性化即时通讯软件。

项目背景：

1. 通过前两年学校课程的学习，我们初步掌握了计算机领域的基础知识及常用算法，并对计算机结构系统的设计有了初步认识，在此基础上，需要通过适当程度的项目开发工作，将学校中学到的知识应用于实践，在实践中发现自己的知识薄弱之处，不断反思提升，并锻炼团队工作能力，为未来的职业生涯打下基础
2. 随着时代的发展，人与人之间的距离逐渐被拉近，人们对于通讯的即时性的要求也越来越高，即时通讯软件也成为业界开发重点，相关领域的积累也较为丰富
3. 在老师的指导下，我们初步掌握了Windows环境下QT的安装和开发技能，并持续学习MySQL和服务器相关知识，为项目的开发打下了较为完善的基础。

项目意义：

1. 通过项目开发，在实战中检验自己的课程学习能力和掌握程度，在实战的思考与反思中提升自己的技术水平
2. 团队项目开发过程中相互学习，相互帮助，锻炼自己的团队合作意识和分工协作能力，积累工作经验
3. 在开发过程中收集相关领域的学习资料，从而接触到业界最前沿的研究工作，拓展自己的视野

1.2 需求分析

人与人之间良好而高效的沟通，是现代社会运行的基础条件。而在社会发展的浪潮之下，社会分工开始变得更加复杂，更加多样化，每个人对于社会交流和信息分享的需求也更加个性化。在这种情况下，针对特定人群的即时通讯软件就体现出其在保护个人隐私，方便信息分享方面的独特优势，，其价值也体现于此。

1.3 运行环境

客户端windows10、服务器linux端、Qt Creator

# 2 项目设计

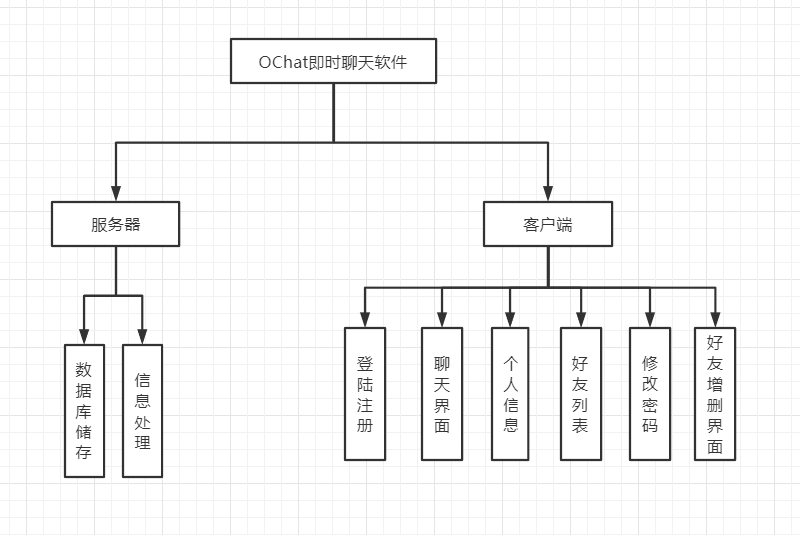
2.1 设计思路

将即时通讯软件命名为OChat，以客户端/服务器（C/S）的模式来设计。客户端负责对话功能，接受用户的输入数据并发送，显示接受的数据；服务器将具体的业务逻辑编入程序中，维护数据库，承担用户的信息储存和消息中转作用。

2.2 模块功能介绍

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **模块** | **功能点** | **功能点详细内容** |
| 1 | 注册模块 | 检查注册用户名 | 如果用户名已注册，提示注册出错 |
| 2 |  | 校验用户输入密码 | 校验用户输入密码是否合法 |
| 3 |  | 传送数据到服务器端 | 将用户填写信息提交到服务器端 |
| 4 |  | 存储到数据库 | 将注册信息存储入数据库，返回注册成功 |
| 5 |  | 修改密码 | 输入用户名、密码向服务器端申请修改密码 |
| 6 |  | 确认密码 | 输入新密码并确认，修改数据库存储密码 |
| 7 | 登录模块 | 校检用户输入内容 | 校检用户输入用户名、密码是否合法 |
| 8 |  | 传送数据到服务器端 | 将用户填写内容提交到服务器端 |
| 9 |  | 返回匹配结果 | 匹配成功返回登录成功，否则返回登陆出错 |
| 10 | 好友模块 | 搜索用户 | 搜索添加好友/删除好友是否存在 |
| 11 |  | 添加好友 | 向服务器发送添加好友请求 |
| 12 |  | 发送好友申请 | 向被添加为好友的用户发送好友申请 |
| 13 |  | 同意好友申请 | 向服务器提交同意好友申请，向数据库添加好友关系 |
| 14 |  | 拒绝好友申请 | 向服务器提交拒绝好友申请 |
| 15 |  | 返回好友申请结果 | 向用户反馈好友申请是否成功 |
| 16 |  | 发送好友列表 | 服务器在用户登陆成功时发送好友列表 |
| 17 |  | 发送好友信息 | 服务器发送所查询好友的填写信息 |
| 18 |  | 删除好友 | 选择好友从好友列表中删除，删除数据库中对应好友关系 |
| 19 | 聊天模块 | 发送消息 | 向服务器提交发送消息和好友 |
| 20 |  | 接收消息 | 服务器将发送的消息发送给好友接收 |
| 21 |  | 存储离线消息 | 服务器检测好友离线将消息暂存 |
| 22 |  | 发送离线消息 | 登陆成功后向用户发送离线期间被发送的消息 |

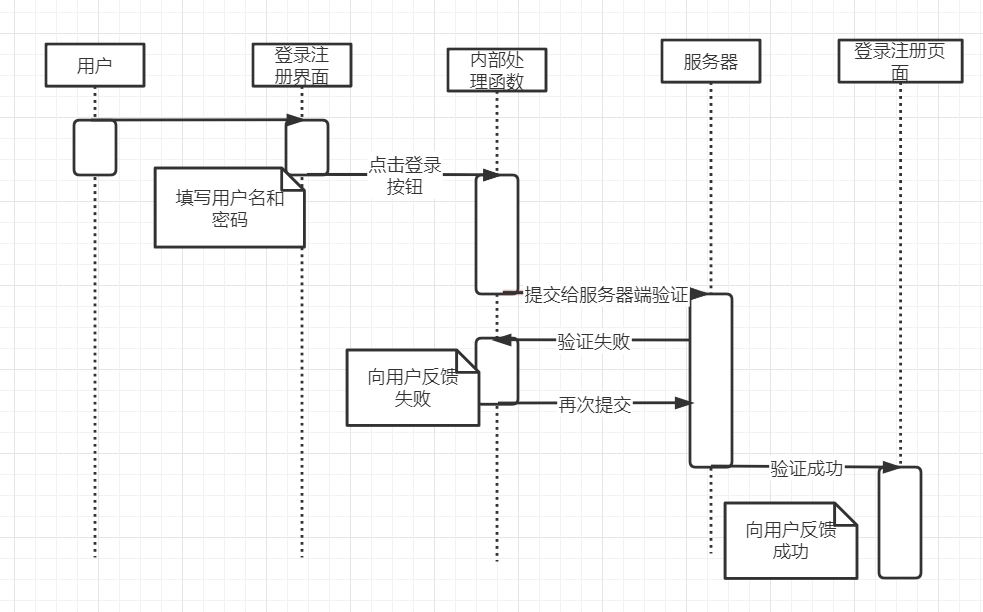
2.3 模块结构图



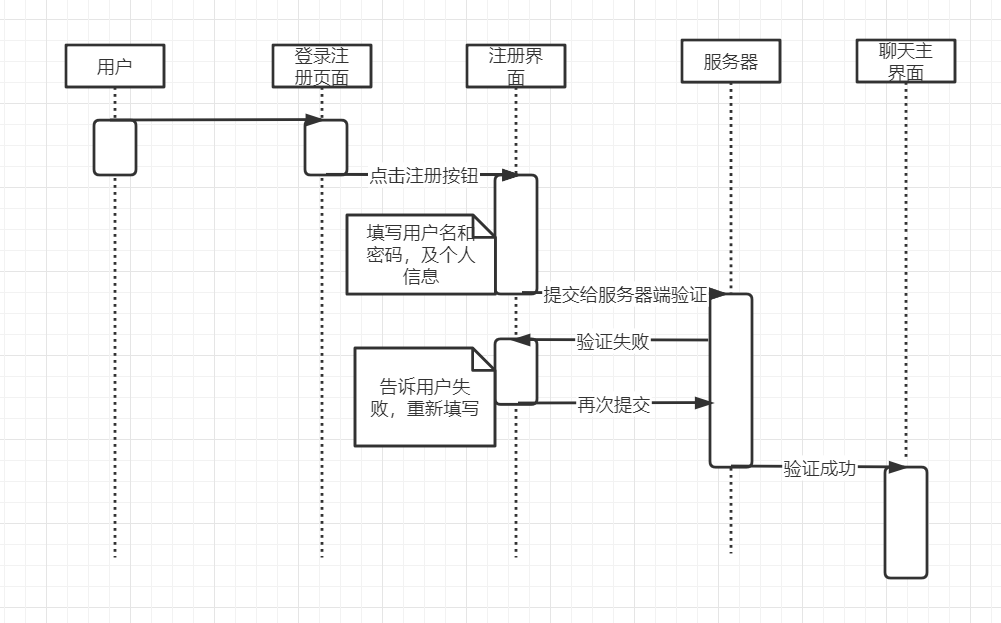
|  |  |  |
| --- | --- | --- |
| 模块名称 | 模块类型 | 概要说明 |
| 登陆界面 | 外部模块 |  |
| 注册界面 | 外部模块 |  |
| 好友增删界面 | 外部模块 |  |
| 聊天界面 | 外部模块 |  |
| 修改密码界面 | 外部模块 |  |
| 个人信息界面 | 外部模块 |  |
| 数据库模块 | 内部模块 | 数据库底层函数设计 |
| Socket模块 | 内部模块 | 数据传输函数设计 |

2.4 程序流程图

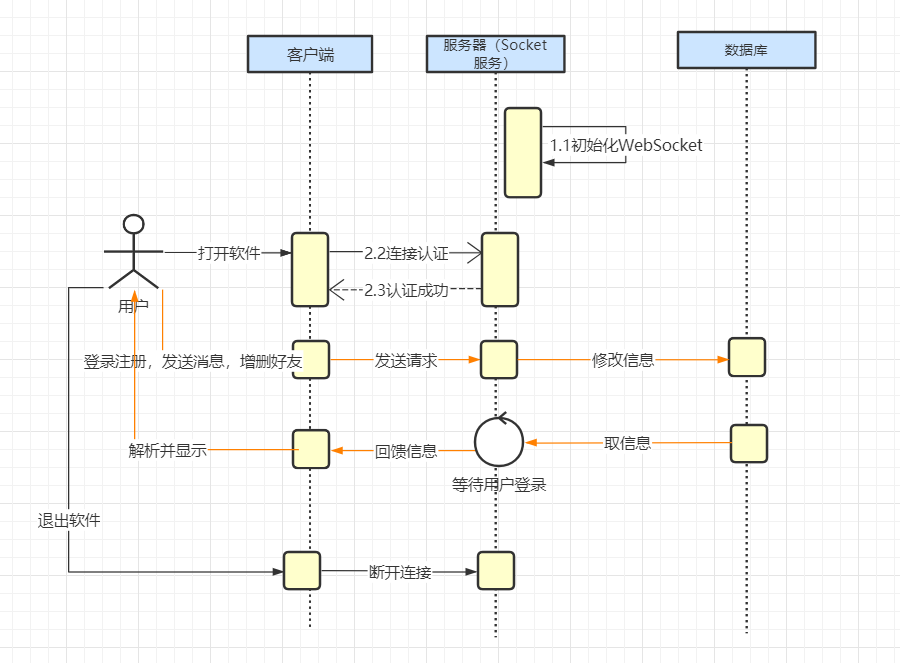
登录时序



注册时序



通信时序图



2.5 功能设计分工

|  |  |
| --- | --- |
| 设计登录注册页面Ui | 刘睿 |
| 添加客户端内的通信交互函数 | 梅浩楠 |
| 主页面聊天窗口多开，头像框 | 梅浩楠 |
| 好友列表及界面布局美化 | 刘睿 |
| 修改密码，添加删除好友UI | 刘睿 |
| 服务器多线程及通信逻辑 | 徐尘化 |
| 数据库建立与维护 | 姜福伟 |
| 局域网通信测试 | 梅浩楠，徐尘化，刘睿 |
| 部分文档编写 | 姚淞瀚 |

# 3 详细设计

**服务器部分：**

服务器设计：服务器端由socket通讯部分、IO复用部分（多并发）、数据库操作部分（MySQL API for C）、客户端消息处理部分以及功能实现函数组成。

其逻辑功能结构如下：

socket通讯部分、IO复用部分、客户端消息处理部分置于主函数int main()中，相互穿插。在socket通讯部分的listen（）函数之后，通过fd\_set和select（）函数、clifd[i]数组进行多并发，每有一个客户端通过连接发来消息，就进入客户端消息处理部分，对客户端消息进行解析，进而根据结果调用各个功能函数，在调用功能函数的过程中，会与数据库进行交互，同时将相应的反馈信息通过send（）函数发送给客户端。

所涉技术：

1. 使用socket进行网络通讯
2. 使用IO复用的select实现多并发
3. 使用sscanf（）函数进行字符串的分解存入
4. 使用MySQL API for C建立与数据库的连接并使用数据库

核心功能：

1. 与客户端进行通讯
2. 同时与多个客户端进行通讯
3. 正确处理客户端的消息
4. 与数据库建立连接并使用数据库
5. 处理用户的不同请求（登录注册、增删好友、发送消息、修改密码、获取好友列表等）
6. 判断用户是否在线以及是否已经离线

**数据库部分：**

user\_tb用户表

user\_name varchar(48), user\_password varchar(16)

存储用户的用户名和密码。

userfd\_tb已登录用户表

user\_name varchar(48), user\_confd int

存储在线用户的用户名和对应句柄。

friend\_tb好友关系表

usera\_name varchar(48), userb\_name varchar(48)

存储用户的用户名和该用户好友的用户名。

message\_tb未读消息表

target\_name varchar(48), message varchar(1024), sender\_name varchar(48)

存储消息内容、发送者和接收者。

firreq\_tb 未读好友申请表

sender varchar(48)(申请发送者), revc varchar(48)（申请接收者）

存储好友申请的发送者和接收者。

accreq\_tb 未读好友申请回复表

req\_target varchar(48)（申请目标/申请接收者）, req\_sender varchar(48)（申请发送者）, attitude char(1)（接收者态度）

存储申请的发送者和接收者和同意/拒绝好友申请的结果。

**客户端部分：**

1. 主体功能界面为登录界面和聊天界面
2. 主要功能界面采用隐藏系统窗口框的方式，自己编写窗口的关闭和最小化界面
3. 在登录界面设置登录注册按钮和修改密码通道
4. 登录打开界面时给出好友列表，用户双击好友开始聊天
5. 当双击好友聊天，好友列表自动向右收缩，在左方显示聊天输入框和好友头像
6. 聊天框支持回车输入和ESC键关闭当前聊天框
7. 在好友列表单击好友可以在右侧查看好友信息
8. 好友列表下方设置添加|删除好友和关闭会话功能按钮
9. 对所有可见按钮和输入框应用样式表进行美化

# 4 实现方案

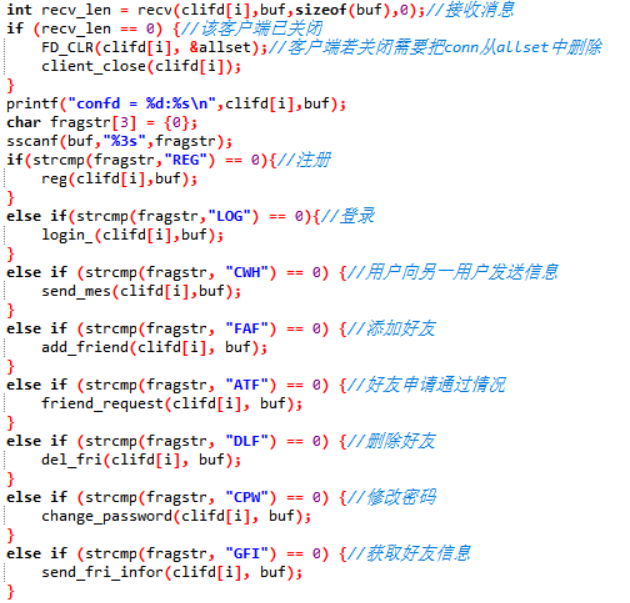
**服务器部分：（徐尘化）**

服务器功能实现方案：

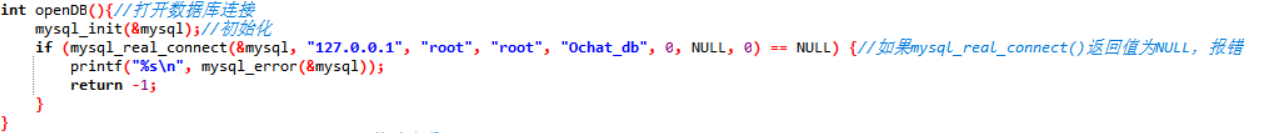
1. 与客户端进行通讯：使用UNIX BSD的套接字（socket）。服务器端先初始化socket(socket()函数)，然后与端口绑定(bind()函数)，对端口进行监听(listen()函数)，调用accept（）函数阻塞，等待客户端连接。在这时如果有个客户端初始化一个socket，然后连接服务器，如果连接成功，这时客户端与服务器端的连接就建立了。客户端发送数据请求，服务器端接收请求并处理请求，然后把回应数据发送给客户端，客户端读取数据，最后关闭连接，一次交互结束。
2. 多并发：使用select IO复用机制。在listen（）函数后，定义两个fd\_set，分别为allset和rset，使用FD\_SET函数首先将服务器套接字放入集合中，然后在一个大循环中反复调用select函数，让我们的程序监视多个文件句柄的状态变化，每有一个句柄连接上就从其那里接收消息。



1. 处理客户端消息：与客户端首先制定好通信协议，使用sscanf函数将客户端发送过来的消息进行分解处理，读出报头来确定客户端是请求什么操作，进而调用对应函数。



1. 使用MySQL API for C中提供的mysql\_init和mysql\_real\_connect函数来初始化数据库并建立连接。使用sprintf函数格式化好所需的sql语句，利用mysql\_query函数来进行查询，调用mysql\_store\_result和mysql\_fetch\_row来获取和逐行读取结果集。



1. 利用设计好的通讯协议，使用sscanf函数将客户端发送过来的消息进行分解处理，读出报头来确定客户端是请求什么操作，进而调用对应函数。然后在对应函数中处理。

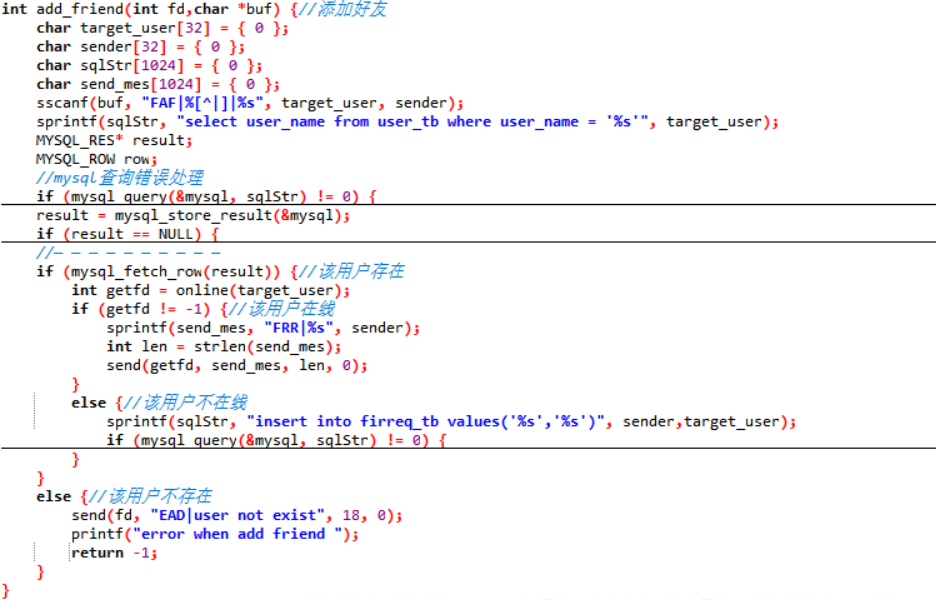
注册：读取客户端发送的注册信息（包括用户名、密码、性别、所在地、生日等），然后将用户名与数据库中存有的名字进行对比，如有重复，则返回客户端错误信息：ERR|用户名已存在。如无重复用户名，则将所有信息存入数据库中。



登录：读取客户端发送的登录信息，首先检查该用户名是否存在，如不存在向客户端返回错误信息，如存在则将该用户名在数据库中保存的密码取出，与发送的密码进行对比，相同则完成登录，将该用户名与该用户名对应的句柄存入数据库；不同则返回错误信息。



添加好友：处理客户端发送的信息，首先检查要添加的用户名是否存在，如不存在向客户端返回错误信息，如存在则将该用户名与发送信息的用户名绑定存入数据库中的好友关系表中。



**数据库部分：（姜福伟）**

user\_tb用户表

在用户使用登陆功能时核对用户输入的用户名和密码，如果用户名与密码对应正确则返回登录成功，否则返回登录失败。

userfd\_tb已登录用户表

用于存储在线用户的用户名，标记每一位用户的在线/离线状态。客户端收到要发送给用户的消息时首先判断用户是否在线，用户在线则直接发送到其客户端，用户离线则暂时存储。在离线用户上线后将消息发送给此用户。

friend\_tb好友关系表

记录用户之间的好友关系，用户的好友列表中会出现所有与其存在好友关系的其他用户，点击好友列表中的好友才能够进行聊天，实现有好友关系的用户之间才可以互相聊天这一功能限制。

message\_tb未读消息表

存储将要发送给离线用户的消息。在记录的离线接收者上线后将存储的消息发送给该用户，并显示所记录的消息发送者。实现发送离线消息的功能。

firreq\_tb 未读好友申请表

存储将要发送给离线用户的好友申请，在记录的离线好友申请接收者上线后向其发送好友申请，并显示好友申请的发送者。实现向离线用户发送好友申请的功能。

accreq\_tb 未读好友申请回复表

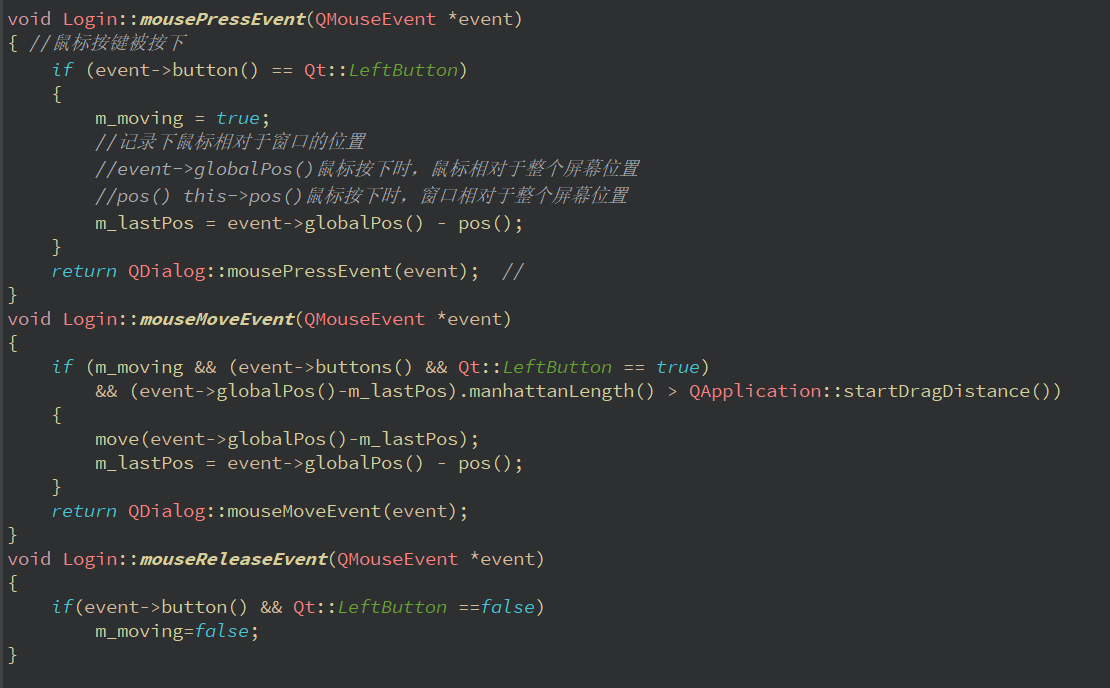
存储将要发送给离线用户的好友申请回复，在记录的好友申请发送者上线后向其发送好友申请回复，告知对方同意/拒绝了好友申请，并显示对方的用户名。在对方同意好友申请的情况下，好友列表会发生改变。实现向离线用户发送好友申请回复的功能。

**客户端部分：（刘睿，梅浩楠）**

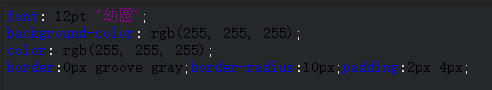
系统窗口框不可见：



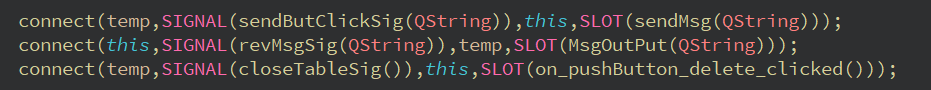
设置不可见后使窗口可拖动：通过设置捕捉鼠标事件和窗口相对位置来实现



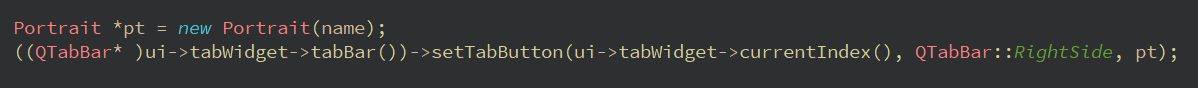
改变控件样式表（以按钮为例）：



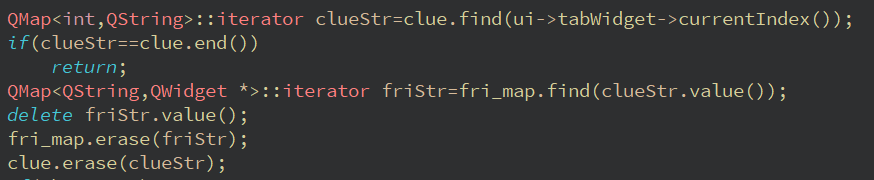
页面之间的数据传输（以初始界面和聊天界面为例）：



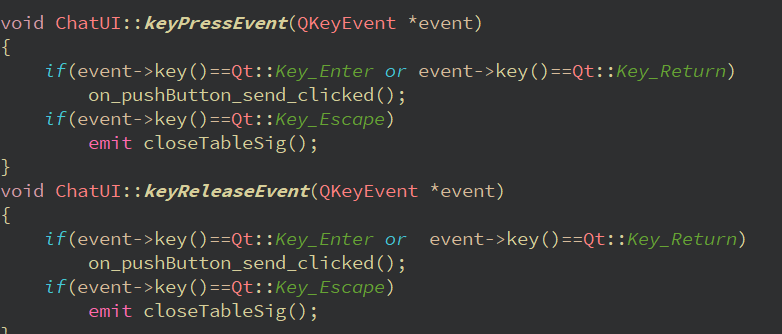
自定义QtableWidget标签：



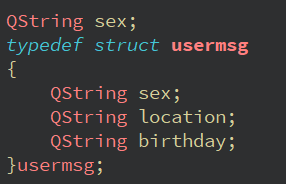
标签的关闭（结束聊天）：



捕捉键盘事件：



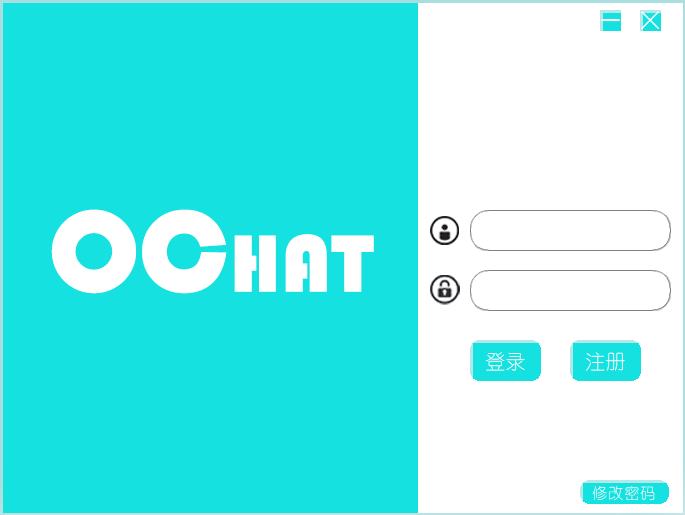
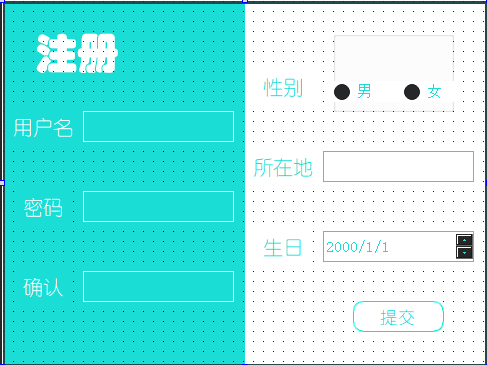
数据结构：



富文本设置文本格式：



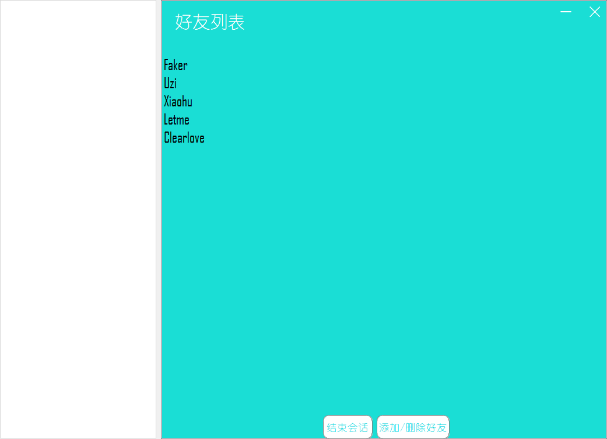
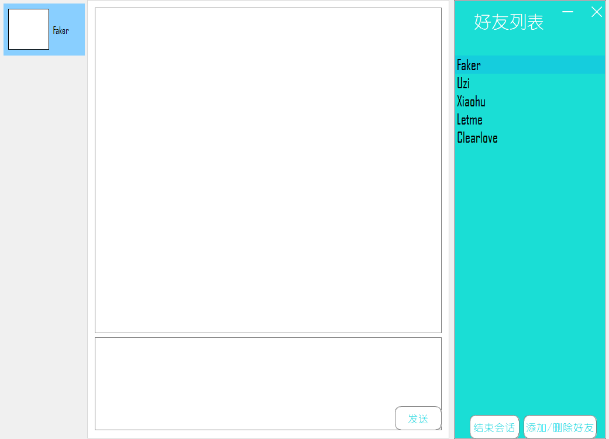
登录界面： 注册界面：

修改密码： 添加/删除好友：

好友列表： 聊天界面：

# 5 系统测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试分类** | **测试内容** | **测试方法** | **判定基准** | **执行結果** |
|
| 注册登录 | 客户端注册/登录界面是否稳定 | 打开客户端 | 客户端能打开并保持稳定 | OK |
| 注册信息能否发到服务器端 | 打开客户端，尝试注册，查看服务器是否接收到消息 | 服务器收到注册信息 | OK |
| 注册信息是否存入数据库 | 打开客户端，输入注册信息，查看服务器是否存入数据 | 服务器存入注册信息 | OK |
| 登录信息能否发到服务器端 | 打开客户端，尝试登录，查看服务器是否接收到消息 | 服务器收到登录信息 | OK |
| 服务器端能否返回注册正确信息 | 打开客户端，输入正确注册信息，查看客户端是否收到注册成功消息 | 客户端收到注册成功信息 | OK |
| 服务器端能否返回登陆错误信息 | 打开客户端，输入错误登录信息，查看客户端是否收到登陆出错消息 | 客户端收到登陆出错信息 | OK |
| 服务器端能否返回登录正确信息 | 打开客户端，输入正确登陆消息，查看客户端是否收到登录正确消息 | 客户端收到登录正确信息 | OK |
| 服务器端能否返回注册错误信息 | 打开客户端，输入错误注册消息，查看客户端是否收到注册出错消息 | 客户端收到注册出错消息 | OK |
| 界面 | 主界面是否稳定 | 正确登录后，是否出现主页面且主页面是否稳定 | 主页出现并保持稳定 | OK |
| 能否正常发送好友列表 | 正确登录，查看客户端能否收到好友列表 | 客户端收到好友列表 | OK |
| 能否发出添加好友申请 | 添加好友，查看在在线/离线情况下是否收到好友申请 | 在在线/离线后上线情况下都可以收到好友申请 | OK |
| 能否发送好友申请同意情况 | 同意/拒绝好友请求，查看在在线/离线情况下是否受到好友申请同意情况 | 在在线/离线后上线情况下都可以收到好友申请 | OK |
| 能否删除好友 | 删除好友，查看好友列表变化 | 删除的好友不出现在好友列表中 | OK |
| 能否接收非好友用户消息 | 向离线好友发送消息后删除好友，好友上线查看消息 | 不能接收到消息 | OK |
| 能否修改密码 | 修改密码，查看服务器是否修改存储密码 | 服务器存储密码修改 | OK |
| 能否查看好友消息 | 点击好友列表中的好友，查看客户端是否收到好友信息 | 客户端收到好友信息 | OK |
| 私聊 | 客户端能否将信息发至服务器端 | 向好友私聊，查看服务器端是否收到消息 | 服务器收到消息 | OK |
| 服务器端能否将一个客户端发送的消息发到另一个客户端 | 向好友私聊，查看好友客户端是否收到消息 | 好友客户端收到消息 | OK |
| 服务器端能否存储离线消息 | 向离线好友私聊，查看服务器端是否存储离线消息 | 服务器存储离线消息 | OK |
| 服务器端能否将离线消息发到登录用户 | 离线用户登录，查看是否收到离线消息 | 登陆后收到离线消息 | OK |