
目 录

| | |
|-----------------------------|----|
| 一、任务描述及设计要求..... | 3 |
| 1.1 设计目的..... | 3 |
| 1.2 设计要求..... | 3 |
| 1.3 设计内容..... | 3 |
| 二、开发环境与工具..... | 3 |
| 三、设计原理..... | 4 |
| 3.1 TCP 通信原理 | 4 |
| 3.2 Socket 通信原理..... | 5 |
| 3.3 HTTP 通信原理..... | 5 |
| 四、系统功能描述及软件模块划分..... | 6 |
| 4.1 系统功能描述..... | 6 |
| 4.2 软件模块划分..... | 8 |
| 五、设计步骤..... | 9 |
| 5.1 基于 Socket 的通信的具体实现..... | 9 |
| 5.2 数据库表的设计 | 11 |
| 5.3 HTTP 通信接口设计..... | 13 |
| 5.4 通信加密设计..... | 14 |
| 六、关键问题及其解决方法..... | 15 |
| 6.1 用户身份验证问题..... | 15 |
| 6.2 实时通讯的并发性问题..... | 16 |
| 6.3 聊天记录的存储和读取问题..... | 17 |
| 6.4 通讯过程中的加密问题..... | 17 |
| 七、设计结果..... | 19 |
| 7.1 服务器设计结果..... | 19 |
| 7.2 客户端各功能结果..... | 19 |
| 7.3 数据库设计结果..... | 22 |
| 7.4 通信接口测试结果..... | 23 |

| | |
|-----------------------|----|
| 7.5 通信加密测试结果..... | 24 |
| 八、软件使用说明..... | 25 |
| 8.1 登录和注册功能..... | 25 |
| 8.2 实时通讯功能..... | 25 |
| 8.3 聊天记录缓存和读取..... | 26 |
| 8.4 表情、图片和文件传输功能..... | 26 |
| 8.5 加密功能..... | 26 |
| 8.6 群聊管理和好友添加..... | 26 |
| 九、参考资料..... | 26 |
| 十、验收时间及验收情况..... | 27 |
| 10.1 工作和验收介绍内容..... | 27 |
| 10.2 老师验收提出的意见..... | 28 |
| 十一、设计体会..... | 29 |
| 11.1 待改进方面..... | 29 |
| 11.2 个人收获部分..... | 29 |

一、任务描述及设计要求

1.1 设计目的

- 1.熟悉开发工具(Visual Studio、C/C++、Java 等)的基本操作;
- 2.了解基于对话框的 windows/Linux 应用程序的编写过程;
- 3.对于 Socket 编程建立初步的概念。

1.2 设计要求

1. 熟悉 Socket API 主要函数的使用;
2. 掌握相应开发工具对 Socket API 的封装;
3. 制作基于局域网的一对一，一对多，多对多的网络即时通讯工具，实现基本数据的网络传输。
4. 基于 Socket API 编程，实现网络消息广播的实用程序。
5. 建议在尝试基本 socket 通信编程后，利用 XAMPP 实现仿 QQ 聊天的快速编程实现。

1.3 设计内容

服务器端设计，客户端设计

二、开发环境与工具

开发环境: Windows10 + Python3.8 + MySQL8.0.26

开发工具: PyCharm Community Edition 2021.3.1

| | |
|--------|--|
| 设备名称 | LAPTOP-4LJFQVNK |
| 处理器 | AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz |
| 机带 RAM | 16.0 GB (15.4 GB 可用) |
| 设备 ID | 15DFAA19-A53C-4B3D-A9BC-49DD1D706B88 |
| 产品 ID | 00342-35844-21972-AAOEM |
| 系统类型 | 64 位操作系统, 基于 x64 的处理器 |
| 笔和触控 | 没有可用于此显示器的笔或触控输入 |

图 2-1：开发系统配置

三、设计原理

在此次课程设计中, 我所完成的课程设计的内容涉及到的原理有三个部分, 一是 TCP 通信原理, 即面向连接的可靠传输层协议; 二是 Socket 通信原理, 即应用层与 TCP/IP 协议族通信的中间软件抽象层; 三是 HTTP 通信原理, 即应用层协议, 它是基于 TCP 协议的应用层传输协议。以下则分别介绍原理和使用方式。

3.1 TCP 通信原理

1、服务器端通信流程（被动接收连接的角色）

(1) 创建一个用于监听的套接字(监听: 监听有客户端的连接; 套接字: 这个套接字其实就是一个文件描述符); (2) 将这个监听文件描述符和本地的 IP 和端口绑定 (IP 和端口就是服务器的地址信息), 其中客户端连接服务器的时候使用的就是这个 IP 和端口; (3) 设置监听, 监听的 fd 开始工作; (4) 阻塞等待, 当有客户端发起连接, 解除阻塞, 接受客户端的连接, 会得到一个和客户端通信的套接字; (5) 通信, 即接收数据和发送数据; 6) 通信结束, 断开连接

2、客户端通信流程(主动发出连接的角色)

(1) 创建一个用于通信的套接字 (fd); (2) 连接服务器, 需要指定连接的服务器的 IP 和端口; (3) 连接成功了, 客户端可以直接和服务器通信, 即开始接收数据和发送数据; (4) 通信结束, 断开连接

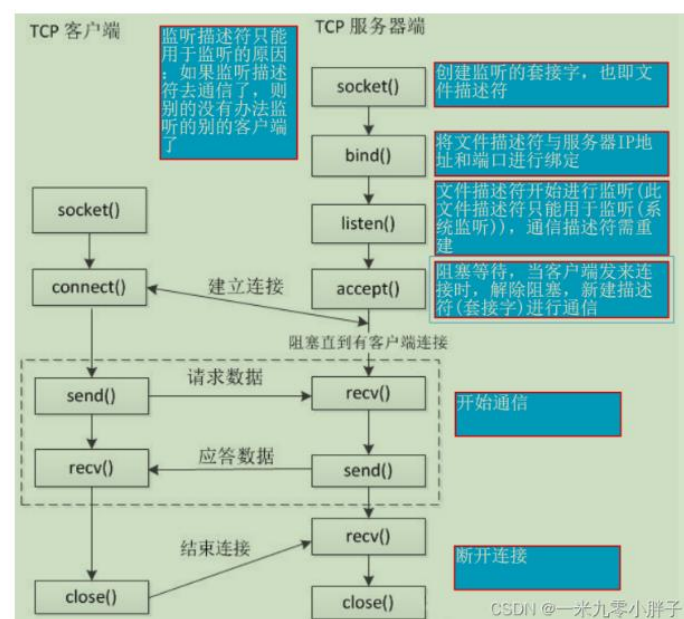


图 3-1 : TCP 通信流程图

3.2 Socket 通信原理

Socket 是应用层与 TCP/IP 协议族通信的中间软件抽象层，它是一组接口，它把复杂的 TCP/IP 协议族隐藏在 Socket 接口后面，对用户来说，一组简单的接口就是全部，让 Socket 去组织数据，以符合指定的协议，对程序员来说，只要用好 socket 相关的函数，就可以完成数据通信。

Socket 起源于 Unix，而 Unix/Linux 基本哲学之一就是“一切皆文件”，都可以用“打开 open -> 读写 write/read -> 关闭 close”模式来操作。可以认为 socket 是一种特殊的文件，一些 socket 函数就是对其进行的操作（读/写 IO、打开、关闭）。相关函数有 bind 函数、listen()函数、connect()函数等。

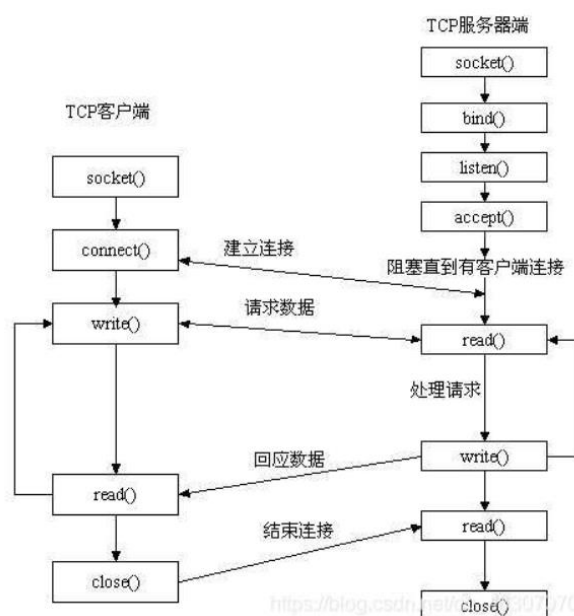


图 3-2 : Socket 通信流程

3.3 HTTP 通信原理

HTTP 协议(超文本传输协议 HyperText Transfer Protocol)，它是基于 TCP 协议的应用层传输协议，简单来说就是客户端和服务端进行数据传输的一种规则。客户端与服务器的角色不是固定的，一端充当客户端，也可能在某次请求中充当服务器。这取决于请求的发起端。HTTP 协议属于应用层，建立在传输层协议 TCP 之上。客户端通过与服务器建立 TCP 连接，之后发送 HTTP 请求与接收 HTTP 响应都是通过访问 Socket 接口来调用 TCP 协议实现。

下图是在网上找的一张图，觉得能很好的表达 HTTP 请求的所发送的数据格

式。由下图可以看到，HTTP 请求由请求行，消息报头，请求正文三部分构成。



图 3-3 : HTTP 请求的所发送的数据格式

请求行由请求 Method, URL 字段和 HTTP Version 三部分构成，总的来说请求行就是定义了本次请求的请求方式，请求的地址，以及所遵循的 HTTP 协议版本；消息报头由一系列的键值对组成，允许客户端向服务器端发送一些附加信息或者客户端自身的信息；请求正文只有在发送 POST 请求时才会有请求正文，GET 方法并没有请求正文。

与 HTTP 请求类似，HTTP 响应也由三部分组成，包括状态行，消息报头，响应正文。下图是其中的结构部分，具体含义则不再解释：



图 3-4 : HTTP 响应的所发送的数据格式

四、系统功能描述及软件模块划分

4.1 系统功能描述

在本次课程设计中，我的主要任务是设计和实现一个实时网络通讯工具。为了达到这个目标，我已经完成了以下功能的实现。

首先，我使用了 Socket 和 MySQL 来实现客户端的登录和注册功能。通过 Socket 的连接和 MySQL 数据库的交互，我实现了对用户的安全的身份验证和注册操作。接下来，我利用多线程技术实现了一对一和一对多的实时通讯功能。借

助多线程的并发处理，可以同时进行私聊和群聊。为了方便用户的使用和管理，我完成了聊天记录的缓存和读取功能的设计，这样方便可以随时查看以往的聊天记录，方便回顾和追溯。

并且为了更好地管理群聊和好友关系，我使用了 Flask 框架来实现群聊管理、好友添加、表情和图片发送等功能。借助这个框架，我可以方便地管理群聊和好友列表，进行各种操作和交流。同时，我还设计了消息的及时显示和处理功能，确保不同的用户能够实时地收到和处理消息。

除了文字聊天，我还实现了简单的文件传输功能。通过 Socket 的传输机制，用户可以快速地发送和接收文件。并且为了保护通信过程的安全性，我采用了 DES 加密算法来实现加密功能。通过这种加密算法，我可以确保通信过程中的数据机密性。同时，为了提高数据的安全性，我还使用了 MD5 加密算法来对用户的信息进行加密处理。

综上，通过我完成的这些功能的设计和实现，我成功构建了一个功能完善且安全可靠的实时网络通讯工具。这个工具可以满足基本的通讯需求，提供了便捷的文字聊天、文件传输和群聊管理等功能。如下图所示为当前针对这一选题所完成的功能思维导图：

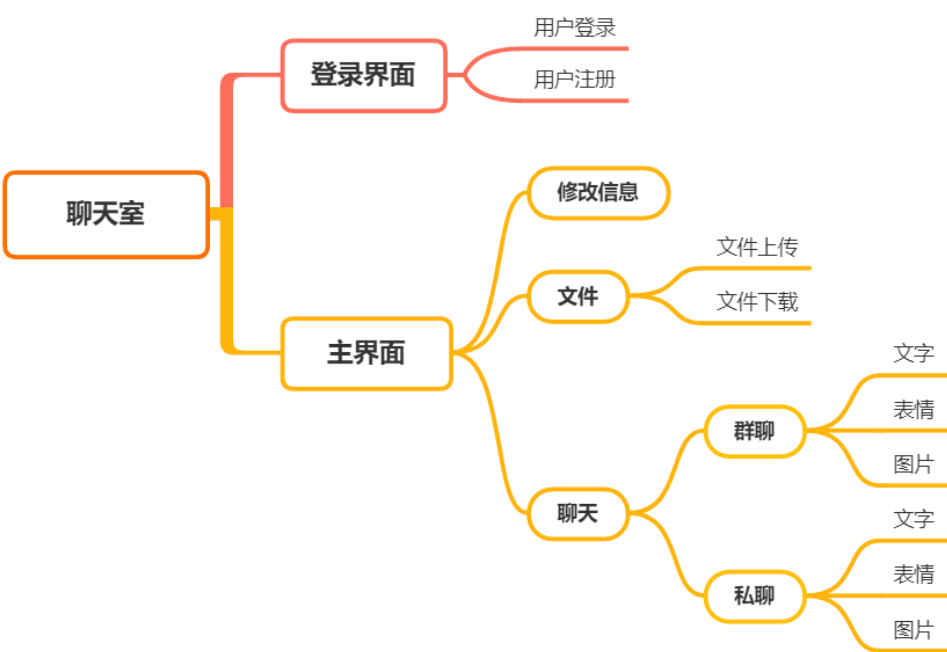


图 4-1：系统整体功能

4.2 软件模块划分

在具体实现的时候，我分成了很多份的代码文件具体实现各部分的功能，每个文件都扮演着不同的角色和功能，共同构成了实时网络通讯工具的设计和实现的重要组成部分。以下是各部分代码文件的文件说明和功能描述：

`ChatWindow.py` 是聊天主窗口框架，该部分代码的功能是实现了聊天主窗口的界面设计和交互逻辑，提供了用户与好友或群组进行聊天的界面。

`FriendView.py` 是好友的私聊视图，该部分代码的功能是基于 `ChatWindow` 框架，展示用户的好友列表，支持与好友进行私聊的功能。

`GroupsView.py` 是群组视图，该部分代码的功能是基于 `ChatWindow` 框架，展示用户所在的群组列表，支持群组内的聊天和交流。

`Home.py` 是用户主界面部分的 UI 代码，该部分代码的功能是展示用户的主界面，包括好友列表、群组列表等，并提供用户进行相关操作的入口。

`Login.py` 是用户的登录界面，该部分代码的功能是展示用户登录的界面，接收用户的登录信息，并进行验证和身份验证。

`Server.py` 是服务器设计和实现的代码，该部分代码的功能是实现了服务器端的逻辑和功能，接收和处理客户端的请求，维护用户信息和通信的连接。

`server.conf` 是服务器配置文件，该部分代码的功能是存储服务器的相关配置信息，如端口号、数据库连接信息等。

`UtilsAndConfig.py` 是服务器与客户端通信设计相关工具文件，因此该部分代码的功能是提供了服务器与客户端通信过程中的各种工具函数和配置参数。

`MyDES.py` 是 DES 加密算法类的实现代码，该部分代码的功能是实现了 DES 加密算法的各种操作，包括密钥生成、加密、解密等。

`DesSetting.py` 是 DES 加密算法框架，该部分代码的功能是提供 DES 加密算法的设置和相关参数配置，用于对通信过程中的数据进行加密。

`DESUtils.py` 是提供 DES 加密算法的相关转换函数，因此该部分代码的功能是提供 DES 加密算法中的工具函数相关的操作。

总体来说，各部分代码各自完成自身的功能，每个文件共同构成了本次课程设计的软件部分，以下是所有文件的汇总统览：












| | | | |
|---|-----------------|-------------|-------|
|  ChatWindow.py | 2023/5/18 23:18 | Python File | 29 KB |
|  DesSetting.py | 2023/6/3 16:06 | Python File | 5 KB |
|  FriendView.py | 2023/5/16 20:54 | Python File | 5 KB |
|  GroupsView.py | 2023/5/14 20:45 | Python File | 3 KB |
|  Home.py | 2023/5/18 23:52 | Python File | 39 KB |
|  Login.py | 2023/5/15 22:08 | Python File | 12 KB |
|  MyDES.py | 2023/5/18 23:06 | Python File | 12 KB |
|  server.conf | 2023/5/13 17:42 | CONF 文件 | 1 KB |
|  Server.py | 2023/5/19 13:08 | Python File | 46 KB |
|  DESUtils.py | 2023/5/18 23:10 | Python File | 4 KB |
|  UtilsAndConfig.py | 2023/5/14 19:58 | Python File | 8 KB |

图 4-2：代码文件部分汇总

五、设计步骤

5.1 基于 Socket 的通信的具体实现

在基于 Socket 实现双方通信的实现部分，我任务最重要的一个部分就是当前的消息协议，因此我对于这一部分进行了详细的设计和实现，如下是消息内容和消息类别的设计思维导图：

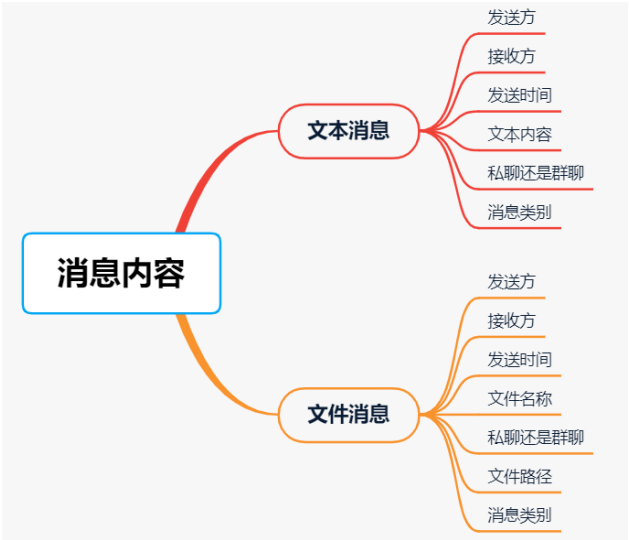


图 5-1：通信消息设计



图 5-2：消息类别设计

如下所示为具体实现中客户端与服务器之间通信的 JSON 消息的具体内容，其中 from_uid 代表发送方的 id 编号(该编号唯一), to_id 代表接收方的 id 编号(该编号唯一), send_time 代表发送时间, file_name 代表发送文件时文件名称, is_friend 用于判断是私聊还是群聊, type 代表消息类别, file_path 代表如果发送文件, 该文件的存储路径, msg_text 代表消息内容, msg_type 代表消息类别。

```
data = {'from_uid': from_id, 'to_id': to_id, 'send_time': send_date, 'file_name': file_name,
        'is_friend': is_friend, 'type': UtilsAndConfig.CHAT_SEND_FILE_SUCCESS,
        'file_path': return_file_path}

data = {'from_uid': from_uid, 'to_id': to_id, 'send_time': send_time, 'msg_text': msg_text,
        'is_friend': is_friend, 'type': '', 'msg_type': 'train'}
```

图 5-3：消息的 json 各个字段内容

最后服务器通过消息类型和消息内容来进行进一步的消息处理，是转发还是返回响应的具体内容给对于的客户端。

而由于 TCP 的可靠性，面向连接特点，因此设计的客户端与服务器之间的通信是基于 TCP 协议的。服务器可以通过 TCP 套接字实时掌握与客户端之间的通信情况。此部分内容既包括消息传输也包括文件传输，对于文件传输则是阻塞型。

其中 Socket 是对 TCP/IP 协议的封装，套接字之间的连接分为三个步骤：服务器监听、客户端请求、连接确认。如下所示为服务器的运行流程图：

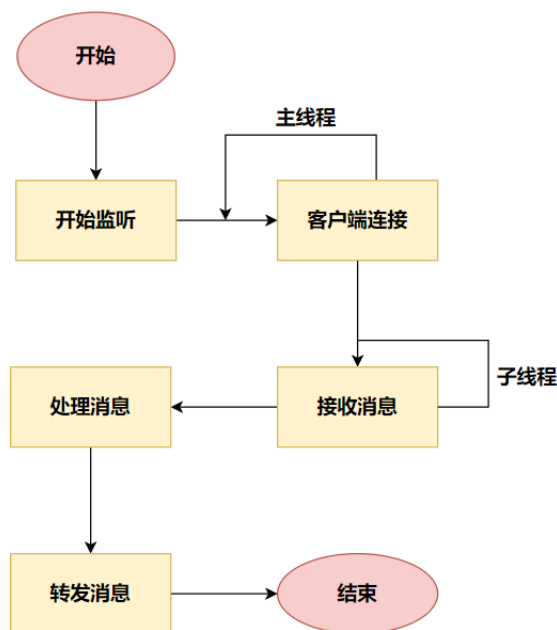


图 5-4：服务器端运行流程图

为了区分不同用户，用户名必须是唯一不可重复的。其中的套接字是服务器发挥中间站转发的关键。其中各个在线的用户的 socket 信息都会存放在一个字典当中，包括 IP 地址、端口号等等，每个子进程就会根据相关信息进行处理，如下图所示。并且为了实现用户主动进行内容的交互，又利用 Flask 框架完成基于 HTTP 的各种查询接口的设计，使得当用户触发某些按钮和界面以后，服务器能从数据库中返回对应的 JSON 数据，进而显示在界面当中。

```
# 连接数据库
db = SQLAlchemy(app)
online_users = dict()
```

图 5-5：利用 dict 保存当前所有的在线用户信息

为了实现通信双方的加密，使用了 DES 加密技术，并约定了对应的口令。而在数据库中使用 MD5 进行加密并且 MD5 不可逆，因此只需比对是否相等即可。

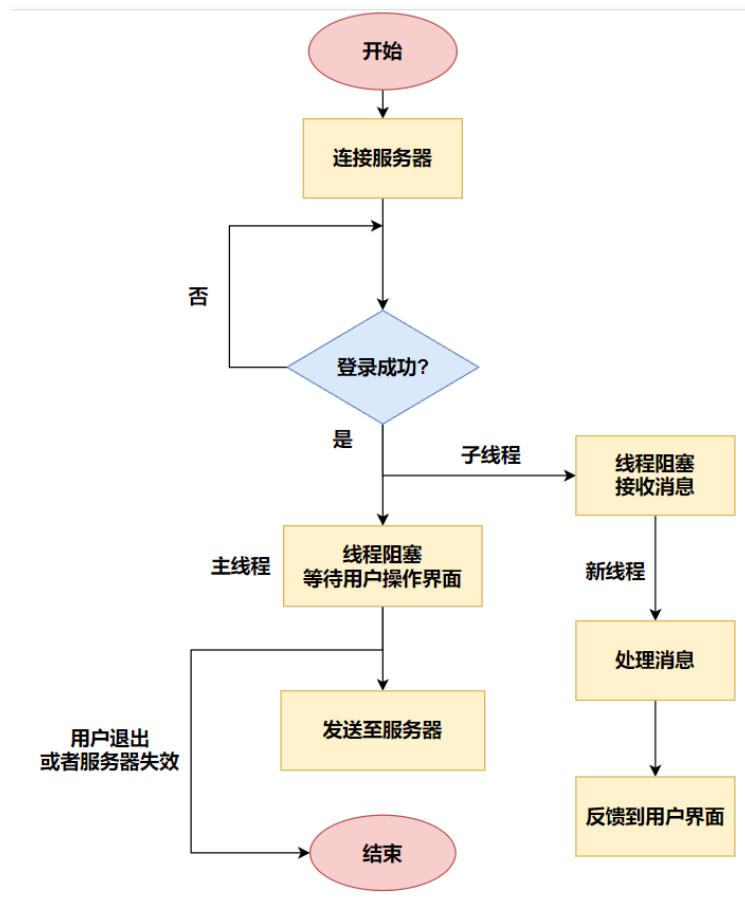


图 5-6：客户端运行流程图

5.2 数据库表的设计

数据库中各表的设计和字段定义是在处理各种操作时最需要考虑的一个部分，为了进一步高效完成这部分内容的实现，我参考了一些开源的博主代码，并基于他们的数据表定义来实现我所需要的信息管理功能，其中一共参考设计了五个数据表：用户表，好友表，群聊表，群聊成员表，聊天记录表，其中表的各个部分表的相关内容如下所示：

用户表设计和字段定义:

```
CREATE TABLE users(  
    id int PRIMARY KEY AUTO_INCREMENT,  
    pwd varchar(36) NOT NULL,  
    loginName varchar(50) NOT NULL UNIQUE,  
    username varchar(50) NOT NULL,  
    sex int not null,  
    picUrl varchar(200) not null,  
    updateTime datetime not null  
);
```

好友表设计和字段定义:

```
CREATE TABLE friends(  
    id int PRIMARY KEY AUTO_INCREMENT,  
    uid int NOT NULL,  
    friendId int NOT NULL,  
    updateTime datetime not null,  
    FOREIGN KEY (uid) REFERENCES users (id),  
    FOREIGN KEY (friendId) REFERENCES users (id)  
);
```

群聊表设计和字段定义:

```
CREATE TABLE groupChat(  
    id int PRIMARY KEY AUTO_INCREMENT,  
    createUid int not null,  
    gname varchar(50) not null,  
    updateTime datetime not null,  
    FOREIGN KEY (createUid) REFERENCES users (id)  
);
```

群聊成员表设计和定义:

```
CREATE TABLE groupMembers(  
    id int PRIMARY KEY AUTO_INCREMENT,  
    gId int not null,  
    uId int not null,  
    joinTime datetime not null,  
    FOREIGN KEY (gId) REFERENCES groupChat(id),  
    FOREIGN KEY (uId) REFERENCES users(id)  
);
```

聊天记录表设计和定义:

```
CREATE TABLE message(  
    id int PRIMARY key auto_increment,  
    fromUid int not null,
```

```
toUid int not null,
type int not null,
gId int null null,
handle int not null DEFAULT 0,
sendTime datetime not null,
handleTime datetime null,
FOREIGN KEY (fromUid) REFERENCES users(id),
FOREIGN KEY (toUid) REFERENCES users(id),
FOREIGN KEY (gId) REFERENCES groupChat(id)
);
```

5.3 HTTP 通信接口设计

Flask 是一个 Python 编写的 Web 微框架，让我们可以使用 Python 语言快速实现一个 Web 服务。因此为了实现用户主动与服务器中数据库进行交互，实现数据的同步，我基于 Flask 完成了相关的 HTTP 接口的设计，如下是通信接口的定义和实现，包括三部分内容：接口名称，接口功能和请求方式，具体设计的内容如下表所示：

表 5-1：通信接口定义

| 接口名称 | 接口功能 | 请求方式 |
|-----------------------|--------|------|
| modifyUser | 修改用户信息 | POST |
| getGroupSize | 获取群聊人数 | GET |
| modifyGroupName | 修改群聊名称 | GET |
| getGnameByGid | 获取群聊名称 | GET |
| getUsernameByUid | 获取用户姓名 | GET |
| searchUserByLoginname | 搜索用户姓名 | GET |
| checkLoginName | 检测登录姓名 | GET |
| register | 用户登录 | GET |
| createGroup | 创建群聊 | GET |
| removeMember | 踢出群聊 | GET |
| deleteGroup | 删除群聊 | GET |
| quitGroup | 退出群聊 | GET |
| getFriendsById | 获取好友信息 | GET |

| | | |
|---------------------|---------|-----|
| delete_friend | 删除好友 | GET |
| addFriendsOrGroup | 添加好友 | GET |
| canAddFriends | 判断是否能添加 | GET |
| messageHandle | 处理消息 | GET |
| getNoHandleMessages | 获取未处理消息 | GET |
| downloadFileSuccess | 文件下载成功 | GET |

5.4 通信加密设计

在实现通信加密部分,我采用了在信息安全技术课上学到的 DES 加密技术和 MD5 加密技术,其中 DES 加密用于传输消息的加密,而 MD5 加密技术用于实现用户信息部分的加密。

DES 是分组长度为 64 位的分组密码算法,密钥长度也是 64 位,其中每 8 位有一个奇偶校验位,因此有效密钥长度为 56 位。DES 算法公开,安全性依赖密钥的保密程度。其中 DES 加密算法可以归纳为三个关键步骤:1. 两次初始置换(初始置换和初始逆置换);2. 子密钥控制下的十六轮迭代加密(也称为乘积变换);3. 十六轮子密钥生成;其中 S 盒是 DES 加密算法中唯一的非线性操作,也是 DES 具有较高安全性的关键所在。以下是 DES 加密算法的流程图:

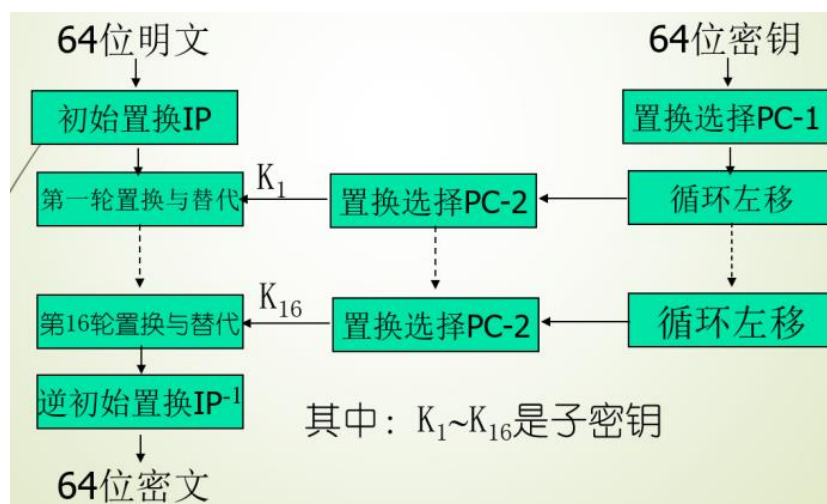


图 5-7 : DES 加密流程

而 MD5 算法为计算机安全领域广泛使用的一种散列函数,用于提供消息的完整性,是计算机广泛使用的哈希算法之一 MD5 的固定长度为 128 比特,16 字节,通常用他的 16 进制字面值输出他,是一个长度为 32 位的字符串。

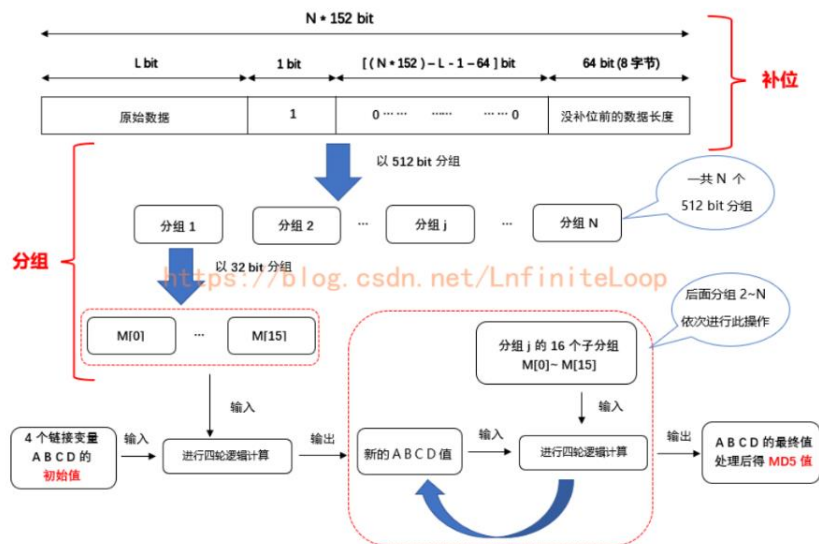


图 5-8 : MD5 加密算法流程图

六、关键问题及其解决方法

在完成本次课程设计实时网络通讯工具的过程中，存在一些关键问题需要解决。以下是这些问题的提出以及相应的解决方法：

6.1 用户身份验证问题

在通信工具的一般实现中有个普遍问题就是如何确保通讯工具只能被授权用户使用，保障通信的安全性和可信性。我的解决方法是采用基于 Socket + MySQL 的用户身份验证机制。通过使用 MySQL 数据库存储用户信息，包括加密的密码和唯一的登录名。当用户在登录时，用户将提供的登录名和密码与数据库中的存储值进行比对。这样，由于 md5 加密算法的不可逆性，只有提供正确的登录凭证的合法用户才能登录和使用通讯工具。

```
# 密码md5加密
md5 = hashlib.md5()
md5.update(pwd1.encode(encoding='utf-8'))
password = md5.hexdigest()

users = Users(pwd=password, loginName=login_name, username=name, sex=sex,
              picUrl=pic_url, updateTime=datetime.datetime.now())
db.session.add(users)
db.session.commit()
```

图 6-1 : md5 加密和使用部分代码

6.2 实时通讯的并发性问题

在具体实现消息的私聊和群聊的时候，我遇到了有两个需要解决的实际问题。一个问题是如何利用 pycharm 实现多客户端的测试，由于代码并不完善，无法将客户端打包成 exe 文件进行多文件测试，因此在查阅相关博客后发现只能另辟蹊径，利用 Pycharm 中的特殊机制实现该功能，需要配置运行方式将代码程序配置为 Python 方式运行，不要使用默认的 Flask 运行。

另一个需要解决的关键问题是如何有效地处理多个用户之间的实时通讯，确保消息传递的稳定性和实时性？我的解决方法是通过多线程实现一对一和一对多的实时通讯功能。每个用户与服务器建立独立的线程，允许并行处理多个用户的消息传递。这样，每个用户可以同时与多个好友或群组进行实时通讯，保证消息能够及时传递和接收。

同时为了实现群聊功能，确保消息能够准确、及时地组播到所有群组成员(非所有用户)，我利用 Flask 框架实现灵活的群聊管理功能。服务器通过创建群组、添加群组成员等操作，实现群聊功能的管理。在群聊过程中，服务器可以将接收到的消息组播给所有群组成员，确保消息能够准确地传递和显示给所有相关用户。

```
# 创建子线程，保持通信
keep_link_thread = threading.Thread(target=socket_keep_link_thread, args=(connection,))
keep_link_thread.setDaemon(True)
keep_link_thread.start()
```

图 6-2：创建子线程继续保持双方通讯

```
# 登录后保持socket连接，处理各种消息的通信
def socket_keep_link_thread(connection):
    while True:
        try:
            msg = connection.recv(1024).decode()
            # print(f"当前msg内容是: {str(msg)}")
            if not msg:
                continue
            else:
                msg_json = json.loads(str(msg))
                # print(f"当前msg_json内容是: {str(msg_json)}")
                # 发消息
                if msg_json['type'] == UtilsAndConfig.CHAT_SEND_MSG:
                    # 发文件
                    if msg_json['type'] == UtilsAndConfig.CHAT_SEND_FILE:
                        # 断开连接，提示好友下线
        except ConnectionAbortedError:
            connection.close(connection)
            return
        except ConnectionResetError:
            connection.close(connection)
            return
```

图 6-3：利用多线程处理各个用户的聊天

6.3 聊天记录存储和读取问题

在处理私聊和群聊的聊天消息的时候考虑到一个问题，如果仅仅只是现场聊天的话那么没有聊天记录缓存也无所谓，但是当涉及到文件发送的时候就需要进行考虑了。

那么此时的问题就变成来了如何高效地存储聊天记录，并提供便捷的方式进行查看和检索，我的解决方法是利用数据库技术，设计并实现专门的聊天记录表。每次用户发送或接收消息时，将消息内容、发送者和接收者的信息、时间戳等关键信息存储到数据库中。这样，用户可以通过查询和读取功能轻松地查看历史聊天记录，包括按照时间、用户等条件进行过滤和检索。

```
# 增加一条聊天记录到缓存中去
def add_one_chat_record(uid, is_friend, from_id, to_id, msg, is_send):
    record_path = get_local_cache_path(uid, is_friend, from_id, to_id, is_send)
    check_path(record_path)
    with open(record_path, 'a', encoding='utf-8') as f:
        f.write('\n' + encode_msg(msg))

# 从本地缓存文件中获取一条聊天记录，并将其解码成字符串。
def get_one_chat_record(is_friend, uid, to_id):
    path = get_local_cache_path(uid, is_friend, uid, to_id, True)
    check_path(path)
    with open(path, 'r', encoding='utf-8') as f:
        lines = f.readlines()
        # 取出最后一行作为聊天记录
        last_line = lines[-1]
    return decode_msg(last_line)
```

图 6-4：聊天记录缓存部分实现函数

6.4 通讯过程中的加密问题

在本学期中我学习了信息安全技术，了解到了其中相关的概念和通信加密技术。因此在本次课程设计中我也考虑到了如何实现高效、安全的文件传输，并保护通信过程中的数据安全性。

在这一部分中，我对这个问题的解决方法是采用 DES 加密算法对数据进行加密，保护通信过程中的数据安全。使用 MD5 加密算法对用户信息进行加密，进一步加强用户隐私的保护。其中 DES 加密算法是不经调用库而自实现的代码，因此在加密时可能在性能方面不如标准库函数。

```

# 利用self.socket.send发送消息
def chat_send_msg(self, is_friend, to_id, send_time, msg):
    send_data = {'type': UtilsAndConfig.CHAT_SEND_MSG, 'isFriend': is_friend,
                 'fromId': self.uid, 'toId': to_id,
                 'sendTime': send_time, 'msgText': msg}

    # 使用des进行加密
    des = MyDES.MyDes()
    text = str(send_data)
    bit64_plain_text_str_array, end_add_zeros_count = utils.group_by_64_bit(text, False)
    if des_setting.IS_PRINT_BIT64_PLAIN_TEXT_STR_ARRAY:
        if end_add_zeros_count > 0 and end_add_zeros_count % 8 == 0:
            bit64_plain_text_str_array, end_add_zeros_count = utils.group_by_64_bit(text, False)
            bit64_plain_text_str_array[-1] = bit64_plain_text_str_array[-1][:end_add_zeros_count:]
    key = "hfut" # 密文
    # 加密
    cipher_text = des.encode(text, key)[0]
    self.socket.send(cipher_text.encode())

```

图 6-5：发送方对发送消息进行加密部分

```

# 保持socket通信
def socket_keep_link_thread(self):
    while True:
        try:
            back_msg = self.socket.recv(1024).decode()
            key = "hfut"
            des = MyDES.MyDes()
            back_msg = des.decode(back_msg, key)
            msg = json.loads(back_msg)
            # 好友状态改变

```

图 6-6：接收方对接收消息进行解密处理部分

```

# 登录后保持socket连接，处理各种消息的通信
def socket_keep_link_thread(connection):
    while True:
        try:
            # 先进行解密
            key = "hfut"
            des = MyDES.MyDes()
            msg = connection.recv(1024).decode()
            msg = des.decode(msg, key)
            # print(f"当前msg内容是: {str(msg)}")
            if not msg:
                continue
            else:
                msg_json = json.loads(str(msg))
                # print(f"当前msg_json内容是: {str(msg_json)}")
                # 发消息
                if msg_json['type'] == UtilsAndConfig.CHAT_SEND_MSG:
                    # 发文件
                    if msg_json['type'] == UtilsAndConfig.CHAT_SEND_FILE:

```

图 6-7：服务器接收方进行解密

如上图所示为利用 DES 加密算法进行加解密的过程，其思路是发送方进行发送前把消息内容进行加密再进行发送，接收方先进行消息解密，提取出关键信息，然后针对消息和文件进行对应处理，并再次加密后发送出去，从而完成整个加解密过程的一个闭环，防止消息在传输过程中被抓包导致泄密。

七、设计结果

7.1 服务器设计结果

服务器的设计思路基于 C/S 架构，旨在用服务器处理客户端的消息。首先，服务器建立与客户端的连接，并监听客户端的请求。然后，服务器接收并解析客户端发送的消息，并根据消息类型进行相应的处理。其中服务器需要实现用户身份验证功能，以验证客户端的登录请求，并根据验证结果进行响应，以及维护用户信息、好友关系和群组信息的数据库，以便进行用户管理和消息转发。

对于一对一聊天，服务器接收来自客户端的私聊消息，并将消息转发给目标好友；对于群聊，服务器接收来自客户端的群聊消息，并将消息以组播的方式发送给群组中的所有成员。服务器还需要处理文件传输请求，接收来自客户端的文件并转发给目标用户或群组。以下是服务器端的主线程部分的代码：

```
# 主线程
if __name__ == '__main__':
    try:
        result = db.session.execute("select 1 as alive")
    except Exception as e:
        sys.exit()
    # 启动socket线程
    socketThread = threading.Thread(target=socket_listen_thread)
    socketThread.setDaemon(True) # 守护线程，这意味着该线程不会阻止程序退出，而是在程序退出时自动结束
    socketThread.start()
    msg = "服务器已启动: " + serverConfig.SERVER_IP + ":" + str(serverConfig.HTTP_PORT)
    # 启动Flask服务器
    app.debug = False
    server = make_server(serverConfig.SERVER_IP, serverConfig.HTTP_PORT, app)
    try:
        server.serve_forever() # 启动一个服务器并一直运行，直到程序退出或手动停止服务器
    # 停止运行时，释放Flask与socket资源
    except KeyboardInterrupt:
        server.server_close()
        mySocket.close()
        sys.exit()
```

图 7-1：服务器主线程部分

7.2 客户端各功能结果

如下是客户端实现的功能结果的截图部分，下图是登录和注册界面：



图 7-2：登录窗口界面



图 7-3：注册账号界面



图 7-4：用户注册

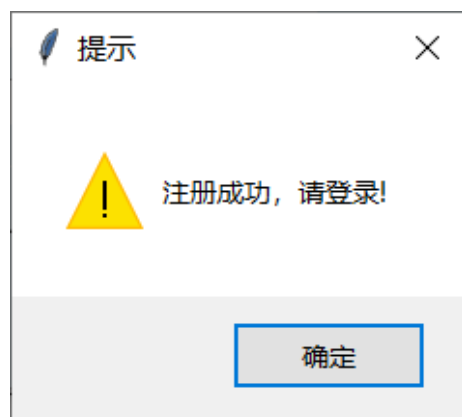


图 7-5：注册结果提醒

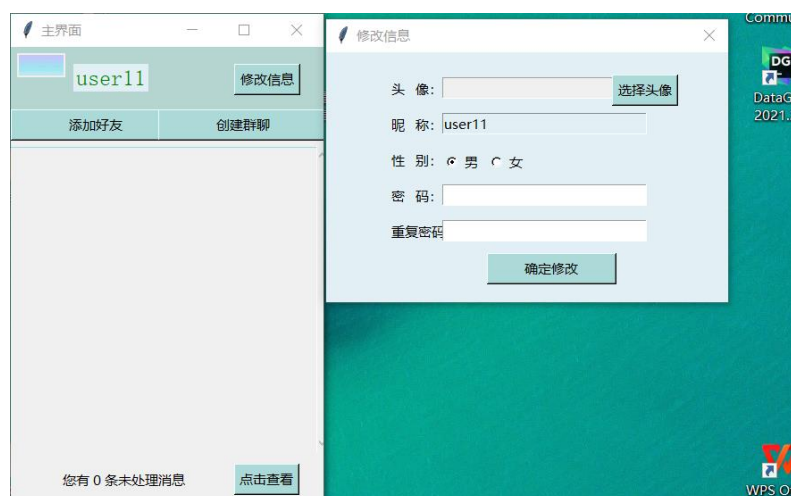


图 7-6：用户登录和修改个人信息界面

如下是聊天主界面和修复信息界面，在主界面会实时现实当前的用户在线情况，并且当有未处理的消息的时候，会进行对应的提示。



图 7-7：聊天主界面

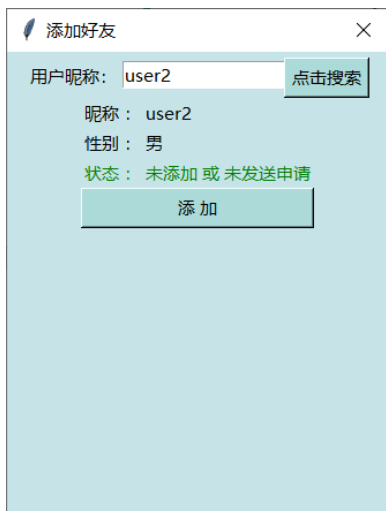


图 7-8：添加好友

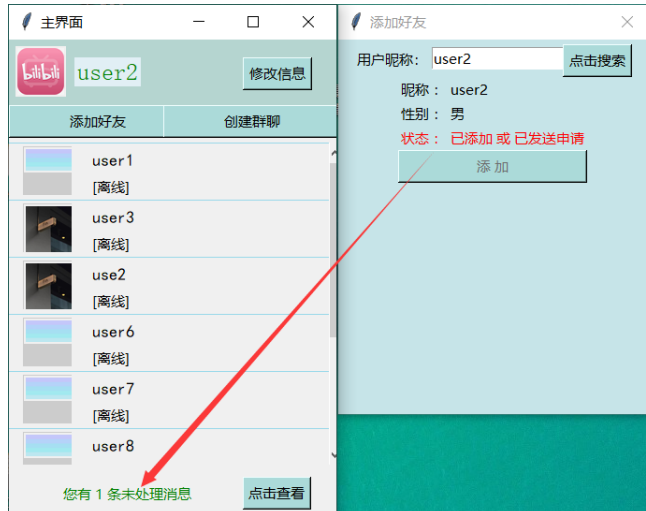


图 7-9：有消息未处理时的结果



图 7-10：消息处理界面

如下是用户与用户之间进行私聊或者群聊部分的结果截图部分，在私聊中用户可以删除好友，群聊中可以退出群聊，并且可以看到当前用户在线情况；并且无论是私聊还是群聊都可以实现各自消息的发送(文本、表情、图片、文件)。在群聊中群主可以对群的相关信息进行处理，包括邀请新人、解散群聊、修改群名。

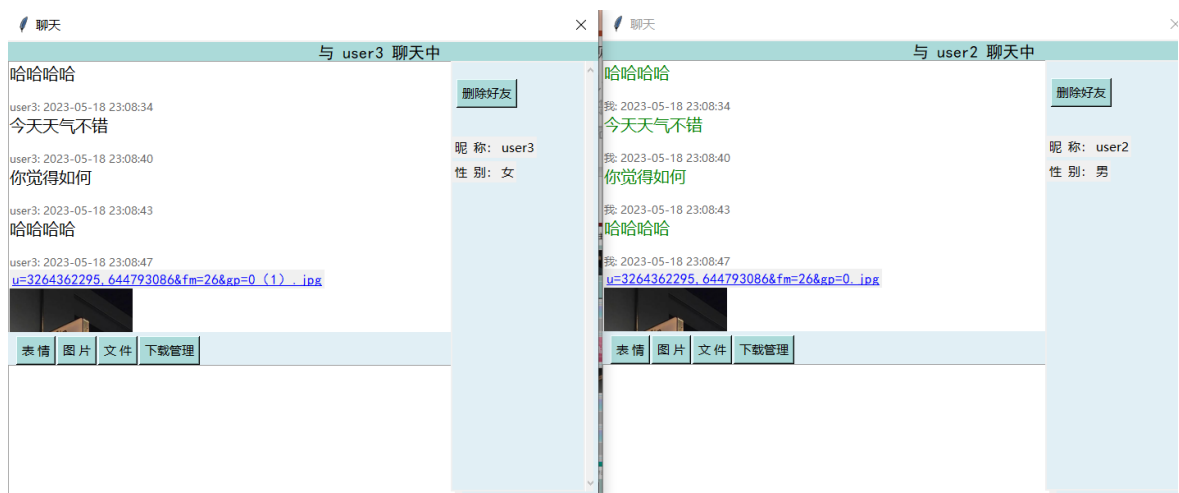


图 7-11：私聊窗口界面



图 7-12 ：群聊窗口界面

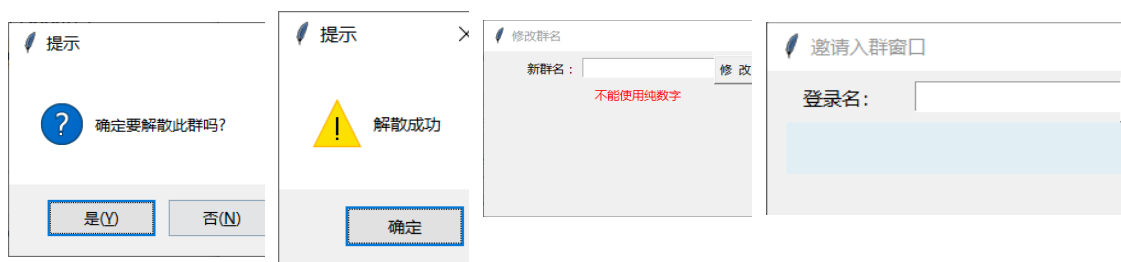


图 7-13 ：群聊管理

如下是用户聊天中的表情发送和文件管理现实，用户可以点击按钮选取对应的表情发送，并且也可以查看当前所有已下载的文件信息。

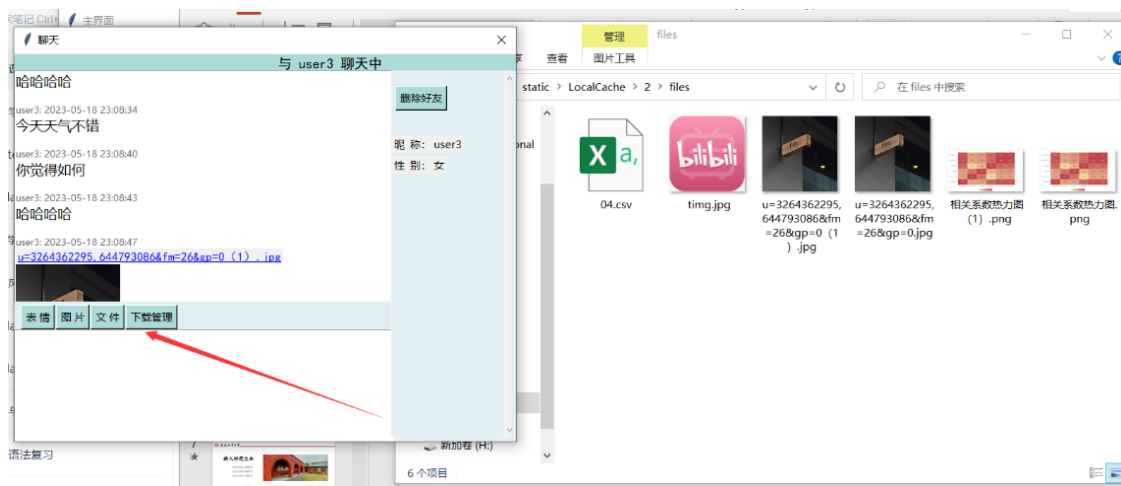


图 7-14: 文件发送和下载管理界面

7.3 数据库设计结果

如下是数据库设计完成所展示的各表的信息关联情况，其中可以清除看到当前表中各个字段的相关信息，包括主码、外键等信息，并且所有信息命名均参照在网上所参考的博主所介绍的命名方式进行命名。其具体内容如下图所示：

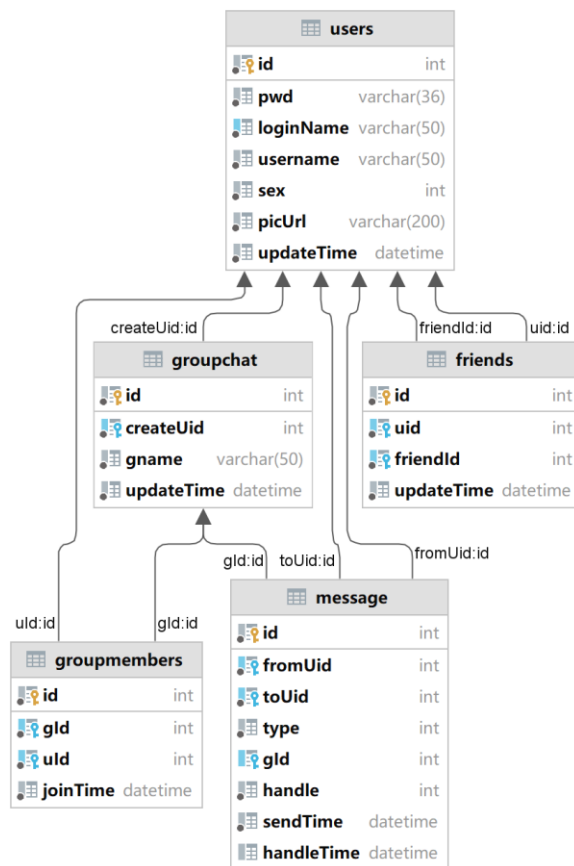


图 7-15：数据库设计结果

7.4 通信接口测试结果

如下图所示为 Flask 相关接口的测试部分结果，由于接口较多，因此只选取了部分接口进行展示，其中所展示的结果是客户端与服务器之间交互的 json 数据，通过各字段数据，客户端和服务端就知道了下一步要进行的操作是什么。




```

127.0.0.1 - - [19/May/2023 11:43:09] "GET /static/headPic/2023-05-09/d0736c53-ee48-11ed-8b44-706655c395f6.jpg HTTP/1.1" 200 11496
127.0.0.1 - - [19/May/2023 11:43:09] "GET /static/headPic/2023-05-09/bffa9b43-ee5c-11ed-bde9-706655c395f6.jpg HTTP/1.1" 200 11496
127.0.0.1 - - [19/May/2023 11:43:09] "GET /static/headPic/2023-05-14/690c29f9-f25a-11ed-8cf1-706655c395f6.jpg HTTP/1.1" 200 25404
127.0.0.1 - - [19/May/2023 11:43:09] "GET /static/headPic/2023-05-14/905d8e0d-f25a-11ed-af88-706655c395f6.jpg HTTP/1.1" 200 25404
127.0.0.1 - - [19/May/2023 11:43:09] "GET /static/headPic/2023-05-14/1b9ef5e7-f25b-11ed-8ddb-706655c395f6.jpg HTTP/1.1" 200 25404
127.0.0.1 - - [19/May/2023 11:43:09] "GET /getGroupMemberCount?gId=1 HTTP/1.1" 200 39
127.0.0.1 - - [19/May/2023 11:43:09] "GET /getGroupMemberCount?gId=2 HTTP/1.1" 200 39
127.0.0.1 - - [19/May/2023 11:43:09] "GET /getGroupMemberCount?gId=4 HTTP/1.1" 200 39
127.0.0.1 - - [19/May/2023 11:43:09] "GET /getGroupMemberCount?gId=5 HTTP/1.1" 200 39
127.0.0.1 - - [19/May/2023 11:43:09] "GET /hasNewMessage?uid=2 HTTP/1.1" 200 39

```

图 7-16：部分通信接口测试结果

7.5 通信加密测试结果

如下是在实现通信加密过程中的测试结果，在此部分展示了 DES 对于一般文本的加密以及发送消息的加密的测试结果，具体内容如下所示：

```

运行: MyDES (1) x
D:\MiniConda_For_Machine_Learning\envs\pyqt5\python.exe F:/Python_Study/multiplayer-chat-room-master/MyDES.py
明文=Cross the stars over the moon to meet your better self. 跨过星河迈过月亮去迎接更好的自己。
明文: Cross the stars over the moon to meet your better self. 跨过星河迈过月亮去迎接更好的自己。
二进制明文(已使用默认utf-8进行编码):
01000011011100100110111011100111001100100000011010001101000
01100101001000000111001101110100011000010111001001110010010000
01101110111011001100101011100100010000001101000110100001100101
0010000001101101011011101101110110111000100000011010001101111
0010000001101101011001010110010111010001000000111100101101111
01110101110010001000000110001001100101011101000111010001100101
011100100010000001110011011001010110110001100110001011100010000
11010001010111101010001101000101111110000111110011010011000
10011111110011010110010101100111110100010111111000100011101000
10111111100001111110011010011100100010001110010010111010101110
1110010110001110101110111101000101111110001101110011010001110
10100101111001101001101110110100111001011010011011110111100111
1001101010000100111010001000011101010111001011011110110110001
11100011100000010000010000000000000000000000000000000000000000
尾部填充了40个0
二进制明文(已使用默认utf-8进行编码)【ps，去除填充得0后得明文编码结果为】:
01000011011100100110111011100110111001100100000011010001101000
011001010010000001110011011101000110000101110010011100110010000
01101110111011001100101011100100010000101110010011100110010000
0110111101110110011001010111001000100000011101000110100001100101
0010000001101101011011101101110111000100000011010001101111
0010000001101101011001010110010111010000100000011100101101111
0010000001101101011001010110010111010000100000011100101101111
01110101110010001000011001001100101011101000111010001100101
解密得到的64bit字符串【ps，去除填充的0】后:
01000011011100100110111011100110111001100100000011010001101000
0110010100100000011100110111010001100001011100100111001100100000
01101110111011001100101011100100010000001110000110100001100101
001000000110110101110110111101110001000000111010001101111
001000000110110110010101100101011101000100000011100101101111
0111010111001000100000110011011001011011000110011000101100010000
11101000101101111010100011101000101111110000111110011010011000
10011111110011010110010101100111110100010111111000100011101000
1011111110000011111001101001110010001000111001001011101010101110
1110010110001110101110111101000101111110001110110011010001110
101001011100110100110111011001110010110100101011110111100111
10011010100001001110100010000111010101110010110111101110001
11100011100000010000010
解密结果为: Cross the stars over the moon to meet your better self. 跨过星河迈过月亮去迎接更好的自己。

```

图 7-17：des 加密原文和编码测试

```

运行: MyDES (1) x
011001010010000001110011011101000110000101110010011100110010000
11101000101101111010100011101000101111110000111110011010011000
10011111110011010110010101100111110100010111111000100011101000
1011111110000111111001101001110010001000111001001011101010101110
1110010110001110101110111101000101111110001110110011010001110
1010010111100110100110111011010011100101101001011110111100111
1001101010000100111010001000011101010111001011011110111010001
11100011100000010000010000000000000000000000000000000000000000
解密得到的64bit字符串【ps，去除填充的0】后:
01000011011100100110111011100110111001100100000011010001101000
0110010100100000011100110111010001100001011100100111001100100000
01101110111011001100101011100100010000001110000110100001100101
001000000110110101110110111101110001000000111010001101111
001000000110110110010101100101011101000100000011100101101111
0111010111001000100000110011011001011011000110011000101100010000
11101000101101111010100011101000101111110000111110011010011000
10011111110011010110010101100111110100010111111000100011101000
1011111110000011111001101001110010001000111001001011101010101110
1110010110001110101110111101000101111110001110110011010001110
101001011100110100110111011001110010110100101011110111100111
10011010100001001110100010000111010101110010110111101110001
11100011100000010000010
解密结果为: Cross the stars over the moon to meet your better self. 跨过星河迈过月亮去迎接更好的自己。

```

图 7-18：des 加密结果和解密结果测试

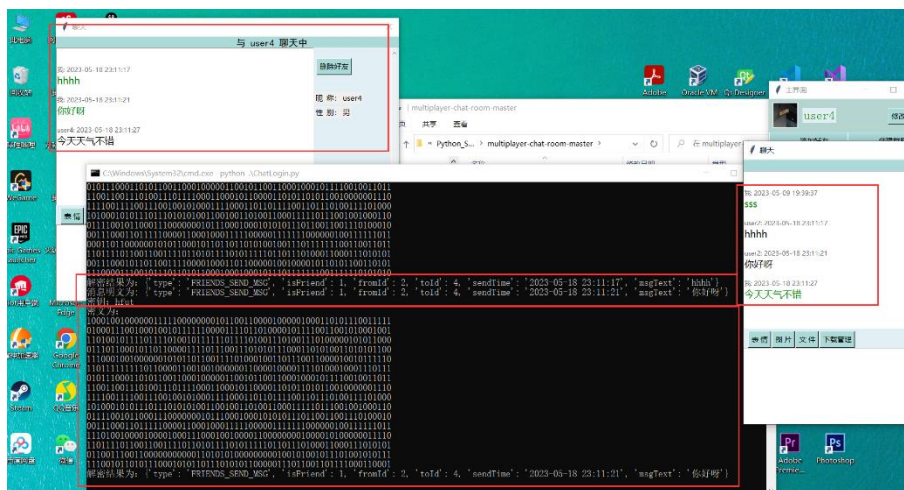


图 7-19：通信双方加密结果测试

| id | pwd | loginName | username | sex | picUrl |
|----|----------------------------------|-----------|----------|-----|---|
| 1 | e10adc3949ba59abbe56e057f20f883e | user1 | user1 | 1 | /static/headPic/2023-05-09/16916af6-ee47- |
| 2 | e10adc3949ba59abbe56e057f20f883e | user2 | user2 | 1 | /static/headPic/2023-05-09/ddbcca71-ee47- |
| 3 | e10adc3949ba59abbe56e057f20f883e | user3 | user3 | 0 | /static/headPic/2023-05-09/d0736c53-ee48- |
| 4 | e10adc3949ba59abbe56e057f20f883e | user4 | use2 | 1 | /static/headPic/2023-05-09/bffa9b43-ee5c- |
| 5 | e10adc3949ba59abbe56e057f20f883e | user5 | xiaowu | 1 | /static/headPic/2023-05-13/ed4a53a9-f172- |
| 6 | e10adc3949ba59abbe56e057f20f883e | user6 | user6 | 1 | /static/headPic/2023-05-14/690c29f9-f25a- |
| 7 | e10adc3949ba59abbe56e057f20f883e | user7 | user7 | 1 | /static/headPic/2023-05-14/905d8e0d-f25a- |
| 8 | e10adc3949ba59abbe56e057f20f883e | user8 | user8 | 0 | /static/headPic/2023-05-14/1b9ef5e7-f25b- |
| 9 | e10adc3949ba59abbe56e057f20f883e | user9 | user9 | 1 | /static/headPic/2023-05-14/2661d752-f25b- |

图 7-20：针对用户密码的 md5 加密结果

八、软件使用说明

8.1 登录和注册功能

在软件启动后，用户将看到登录界面。如果用户已经拥有账户，可以输入用户名和密码进行登录。如果是新用户，请点击注册按钮进行账户注册。注册时，需要提供有效的用户名和密码，并按照要求填写其他必要信息，包括用户头像、性别等。注册成功后，用户可以使用账户进行登录。

8.2 实时通讯功能

本次课设所完成的软件支持一对一和一对多的实时通讯。通过多线程的技术支持，用户可以同时进行私聊和群聊。在软件登录后，用户将看到好友列表和群组列表。双击想要私聊的好友或选择想要进入的群组，即可打开聊天窗口。

在聊天窗口中，用户可以输入文本消息并按下发送按钮发送给对方。私聊好友将收到消息提醒，并及时显示用户发送的消息。在群聊中发送的消息将被组播

到群组的所有成员，其他好友也会收到消息提醒，确保了信息的实时传递和处理。

8.3 聊天记录缓存和读取

该软件支持聊天记录的缓存和读取。用户可以随时查看以往的聊天记录。在聊天窗口中，滚动窗口将显示较早的消息记录。如果用户需要查看更早的记录，只需向上滚动即可。这样，用户可以方便地回顾和追溯与好友或群组的聊天内容。

8.4 表情、图片和文件传输功能

该软件支持简单的文件传输功能。通过选择文件并点击发送按钮，用户可以快速地将文件发送给好友或群组。文件传输过程是基于 Socket 的实现，确保了快速而可靠的文件传输。为了丰富聊天内容，我设计实现了支持表情和图片的发送功能。用户可以在聊天窗口中选择并发送各种表情和图片，与好友或群组分享。

8.5 加密功能

为了保护用户数据的安全性，防止通信过程中被其他人抓包通信信息，我实现了通信的加密功能。通信过程中的数据采用了 DES 加密算法进行加密，确保了通信的机密性。而用户信息采用了 MD5 加密算法进行加密处理，提高了用户数据的安全性。

8.6 群聊管理和好友添加

在设计实现的时候，我基于 Flask 框架实现了群聊管理和好友添加功能。用户可以方便地创建和管理群聊，并邀请好友加入。同时，用户可以通过搜索和添加好友的方式扩展该用户的好友列表，并且所有信息都会及时更新到数据库中进行保存。

九、参考资料

- [1] 谢钧,谢希仁. 计算机网络教程[M].人民邮电出版社:, 201409.350.
- [2] 章全. 计算机网络原理与应用[M].南京大学出版社:, 201912.249.
- [3] 程保中,张笑燕,孙艺,谢锦. 移动互联网软件开发实验指导[M].人民邮电出版社:, 201309.309.
- [4] 唐强.基于 C/S 架构的 QT 局域网通信工具设计[J].计算机光盘软件与应

用,2012,15(21):191-192.

- [5] 孟金红.局域网通信工具的设计[J].中国新通信,2012,14(20):60.
- [6] 黄伟.DES 加密算法的改进方案[J].信息安全与通信保密,2022(07):100-105.
- [7] 马振东,郑永,吴军辉等.基于 socket 通信的多加油站集中信息化管理系统[J].中国石油和化工标准与质量,2017,37(10):57-60.
- [8] 陈香凝,王烨阳,陈婷婷,张铮. Windows 网络与通信程序设计[M].人民邮电出版社:, 201703.477.
- [9] 陈娟.基于 TCP/IP 的局域网通信系统的设计与实现[J].信息与电脑(理论版),2018(22):179-182.
- [10]杨成义,陈科宏.一款局域网通信软件的设计与实现[J].现代信息科技,2020,4(19):81-83.DOI:10.19850/j.cnki.2096-4706.2020.19.020.

十、验收时间及验收情况

10.1 工作和验收介绍内容

在工作方面,本次课程设计我没有选择组队,而是选择了自行设计和开发,从界面到数据库的设计和应用也参考和学习了很多博主的开源代码和博客,并且也简单实现了通信双方的加密环境,因此工作量也不小,花了我很大精力。

而在本次课程设计验收时,老师一共验收了两部分内容,包括代码实现和软件功能。在软件功能部分,我为老师演示介绍了如下功能:

1. 用户登录信息验证和新用户注册信息的功能;
2. 用户与用户之间私聊和群聊的聊天功能(文本、表情、图片、文件);
3. 用户修改个人信息,用户之间相互添加好友和删除好友的功能;
4. 群主进行管理群聊(修改群聊信息、管理群成员)和邀请好友加入群聊的功能;
5. 文件发送和管理以及聊天记录缓存的相关功能的展示
6. 发送消息后用户界面信息提示的功能(处理好友申请、添加群聊、未读消息);

在代码实现部分,我为老师讲解了我基于 Socket 和 Flask 实现用户私聊和群聊功能的相关思路、数据库设计思路、聊天信息转发和处理部分的思路以及通信加密过程中所涉及到的 DES 加密算法的相关实现思路。

10.2 老师验收提出的意见

针对我所演示的功能，老师特别指出了我利用 DES 加密算法在通信加密过程中的不足，让我对通信加密的过程进行对应的改进，因此我后面又对这部分的代码进行了修正，最后实现了所需要的效果。如下是我对客户端发送消息、接收消息和接收端转发消息的代码修正部分：

```
# 利用self.socket.send发送消息
def chat_send_msg(self, is_friend, to_id, send_time, msg):
    send_data = {'type': UtilsAndConfig.CHAT_SEND_MSG, 'isFriend': is_friend,
                 'fromId': self.uid, 'toId': to_id,
                 'sendTime': send_time, 'msgText': msg}

    # 使用des进行加密
    des = MyDES.MyDes()
    text = str(send_data)
    bit64_plain_text_str_array, end_add_zeros_count = utils.group_by_64_bit(text, False)
    if des.setting.IS_PRINT_BIT64_PLAIN_TEXT_STR_ARRAY:
        if end_add_zeros_count > 0 and end_add_zeros_count % 8 == 0:
            bit64_plain_text_str_array, end_add_zeros_count = utils.group_by_64_bit(text, False)
            bit64_plain_text_str_array[-1] = bit64_plain_text_str_array[-1][:end_add_zeros_count:]
    key = "hfut" # 密文
    # 加密
    cipher_text = des.encode(text, key)[0]
    self.socket.send(cipher_text.encode())
```

图 10-1：发送方消息加密部分

```
# 保持socket通信
def socket_keep_link_thread(self):
    while True:
        try:
            back_msg = self.socket.recv(1024).decode()
            key = "hfut"
            des = MyDES.MyDes()
            back_msg = des.decode(back_msg, key)
            msg = json.loads(back_msg)
            # 好友状态改变
```

图 10-2：接收方接收消息处理部分

```
# 登录后保持socket连接，处理各种消息的通信
def socket_keep_link_thread(connection):
    while True:
        try:
            # 先进行解密
            key = "hfut"
            des = MyDES.MyDes()
            msg = connection.recv(1024).decode()
            msg = des.decode(msg, key)
            # print(f"当前msg内容是: {str(msg)}")
            if not msg:
                continue
            else:
                msg_json = json.loads(str(msg))
                # print(f"当前msg_json内容是: {str(msg_json)}")
                # 发消息
                if msg_json['type'] == UtilsAndConfig.CHAT_SEND_MSG:
                    # 发文件
                    if msg_json['type'] == UtilsAndConfig.CHAT_SEND_FILE:
```

图 10-3：服务器接收方解密并转发处理消息

十一、设计体会

11.1 待改进方面

- 在测试大文件的时候, 软件由于没有设计缓冲区会时长崩溃卡死; 并且未考虑文件传输过程中丢包问题, 传输过程中的带宽和并发, 以及大文件传输多线程优化问题, 所以一个重要优化点就是这里;
- 没有涉及到 HTTP 端的 cookie 和 session 处理, 后面可能会针对这部分进行对应的处理进而确保客户端和服务端之间的通信的有效性; 同时界面部分和功能部分还有些 bug 没有来得及调整, 比如消息传输后会被界面挡住等;
- 在实现双方加密通信时候没有考虑周期, 尤其是文件传输的时候, 未考虑文件传输过程中加密对于文件内容的影响, 以及没有对文件传输是否因为加密导致丢包问题进行测试, 未涉及文件验证机制;
- 后面会考虑将后端 server 部分代码和数据库配置到云服务器, 然后将客户端代码部分封装成一个 exe 文件, 进而实现如同 QQ 或者微信一样的个人版在线聊天软件功能, 实现通信工具开发的最后一公里;

11.2 个人收获部分

总体收获:

在网络通信方面, 对 Socket 编程和 Flask 编程方面有了一定的了解和认识, 通过视频学习了解了使用方式, 并对 python 中的多线程技术有了一定的认识; 并且对 Python 中的 Tkinter 这个 GUI 库有了更加深入的使用体验, 特别是设计界面布局和研究各种槽函数的实现;

在数据库的设计和实现方面, 通过学习其他博主基于 SQLAlchemy 实现数据库的操作, 对 Flask 框架下的 SQLAlchemy 有了一些了解, 并且了解并具体实现了如何基于 MySQL 来利用该库实现对数据的增删查改业务;

在网络通信加密方面, 通过学习和使用 DES 加密和 md5 加密技术, 在一定程度上解决了网络抓包导致信息泄露的问题, 同时自身也对网络通信之间双方通信信息的加密处理的意义有了深刻的认识;

在解决问题方面, 查阅了很多很多的博客, 在设计、实现和运行各个部分都解

决了很多问题;同时对一些不常用的库有了一些认识,解决了一些问题,比如使用 uuid 中的库函数来对文件进行唯一的标识符来处理文件同名问题,又比如使用 PIL 库解决图像大小问题,urllib 库解决 URL 在特殊字符的编码处理;

其他感悟:

总的来说,这一次针对实时网络通讯工具的设计和实现的经历对我而言是一次非常新奇,同时也拓宽视野的学习经历,它让我接触到了一些崭新的技术与概念,也让我了解了一些关于网络通讯的具体的运作原理和实现方式,对于我们未来的个人发展有很大的意义。

与此同时,不可否认网络通信是计算机整个体系中非常重要的一个部分,对它进行深入地了解具有重要的意义。在自己完成这次课程设计之前,我对通信这个方向还不是很了解,只是停留在书本知识和老师所讲的 ppt 中。同时这是我首次如此深入地进行软件的设计和实现,经过这次课程设计,我巩固了已有的计算机网络课程知识,也对 Socket 通信机制更加熟悉。

但目前而言,还是有一些不足之处尚待提高,我将通过本次课程设计过程中得到的经验和教训进行查漏补缺,以更积极的态度努力学习。并且值得一提的是,我特别感谢周围同学和网上的一些大佬,在学习的过程中,我发现关于网络通信的具体实现网上有很多人分享自己的学习笔记,所以通过对他们笔记的学习,我也收获了不少的知识。同时和在周围同学的讨论后知道了我上课没有听懂而导致编写代码产生的问题也能迎刃而解。

最后,衷心的感谢计算机网络的周健老师,通过周老师我对计算机整个的网络结构有了更加深入的认识,因为之前也学习过一些相关的知识,但是从整体上来说仍然比较狭义。而通过这门课程设计,我对它又有了更加深刻的认识。并且通过这次的课程设计,我懂得了什么是思考、分享和坚持。并且,也十分感谢各位计算机网络验收的老师,从课设开题到验收结束,老师先详细讲解了各个选题让我清楚了各个题目的要求,并且在验收的时候也提出了一些中肯的意见,让我知道了我目前存在的不足以及可改进的方面。

简而言之,经历完本次课程设计的实现以及课设报告的撰写的这两个过程也算收益颇丰,不虚此行了。